Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Hai Cu

Prediction of forest variables and its reliability with a hyperspectral deep learning model

Master's Thesis
Espoo, July 30, 2021

Supervisor: Jorma Laaksonen D.Sc. (Tech.)
Advisor: Jorma Laaksonen D.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Hai Cu		
Title:	Prediction of forest variables and its reliability with a hyperspectral deep learning model		
Date:	July 30, 2021	**Pages:**	63
Major:	Machine Learning, Data Science and Artificial Intelligence	**Code:**	SCI3044
Supervisor:	Jorma Laaksonen D.Sc. (Tech.)		
Advisor:	Jorma Laaksonen D.Sc. (Tech.)		

With three-quarters of the land surface area is covered by forests, Finland is the most heavily-forested country in Europe. Forests have been the major source of both social well-being and economic of Finland. Therefore, it is important to understand the forest development, vegetation features and quality to help preserve the biological diversity of forests while securing different economic uses of forest in a good manner. In order to understand the forest development, the hyperspectral remote sensing has been used with the artificial intelligence methods to predict and analyze the forest variables. A novel convolutional neural network has been developed in the AIROBEST project to predict various the forest variables simultaneously. Despite of decent results for both categorical and continuous variables, there are a few problems with the predictions obtained from the model.

This thesis tackles the existing challenges in the prediction of forest variables from hyperspectral images by suggesting a new procedure for data processing and data split to utilize the large amount of hyperspectral data. Based on the new data split, many experiments were carried out to create a new baseline model. To review and assess the predictions of the baseline model, new evaluation metrics are proposed for stand-level analysis of the predictions. The predictions are also studied and quantified to understand their variation and reliability.

Overall, the new baseline model achieves a good overall accuracy 83.48% and mean class accuracy 75.13% for the categorical variables. For the continuous tasks, the model reaches 17.24% relative root mean square error for four main continuous variables with a high coefficient of determination 0.75. At the end, the shortcomings of this thesis work are discussed and suggestions are given to improve the results as well as the reference data.

Keywords:	Hyperspectral image, deep learning, convolutional neural network, remote sensing, multi-task learning.
Language:	English

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor and advisor, Dr. Jorma Laaksonen, for his guidance and support during my thesis work. His feedback on my thesis is much valuable and I have learnt a lot from him.

Secondly, I would like to thank M.Sc. Phu Pham for introducing me to the project and assisting in the early phase of my thesis work.

Additionally, my appreciation also extends to the Artificial Intelligence for Retrieval of Forest Biomass and Structure (AIROBEST) team including Dr. Matti Mõttus, M.Sc. Eelis Halme, and M.Sc. Mathiew Molinier for their advice and support. It has been an enjoyable working experience with you all.

Last but not least, I would like to offer my special thank to my cat Minh, girlfriend Ly, family and friends for their emotional support. I would not have done this without you.

Helsinki, July 30, 2021

Hai Cu

Abbreviations and Acronyms

AI	Artificial intelligence
BN	Batch normalization
CI	Confidence interval
CNN	Convolutional neural network
FC	Fully-connected
GPR	Gaussian process regression
HSI	Hyperspectral image
ICA	Independent component analysis
LAI	Leaf area index
LDA	Linear discriminant analysis
MSE	Mean square error
MTL	Multi-task learning
PCA	Principle component analysis
R^2	Coefficient of determination
rBias	Relative bias
ReLU	Rectified Linear Unit
RMSE	Root mean square error
rRMSE	Relative root mean square error
SVR	Support vector regression

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The boreal forest, also called Taiga, is the biome of coniferous forests including various types of pines, spruces and larches. The boreal forest covers 17 million square kilometres, about 30% of the total forest area on the planet [17]. It plays a vital role in Earth's climate system by producing a significant amount of O_2. However, with the climate changes, boreal forests are one of the most affected ecosystems. Therefore, it is critical to ensure the sustainability and well-being of the boreal forests and their ecological values.

Artificial Intelligence for Retrieval of Forest Biomass & Structure, also known as AIROBEST, is a joint research project by the Land Remote Sensing team of VTT Technical Research Center of Finland and the Department of Computer Science at Aalto University and funded by the Academy of Finland. The project aims to study the carbon balance, future climate and the sustainability of the current politics and the emerging bio-economy of the boreal forest. To be able to study and monitor a large surface of the Earth, remote sensing technology is applied to acquire information on the forest from above without physical contact. In the project, a hyperspectral image of a forest region was taken in Southern Finland, near Hyytiälä forestry field station, by an airborne flight campaign for research purposes. The image size is approximately 40 gigabytes of memory and a resolution of 12143×12826 pixels and 128 spectral bands.

Artificial Intelligence dataset for forest geographical applications, also known as TAIGA, consists of the afore mentioned hyperspectral image, its stand-level reference data and the stand boundaries. The reference ground truth of TAIGA has been gathered from the Finnish forest resource data, which is provided by the Finnish Forest Center[1]. The forest resource data is open sourced in 2018 and published under Creative Commons Attribution

[1] `metsaan.fi/paikkatietoaineistot`

4.0 International (CC BY 4.0) license. According to the Finnish Forest Center, the forest data is modeled with growth models and archived with good accuracy such as ± 2 m for mean height and ± 3 m^2/ha for basal area [13]. No changes to the reference data were made in this thesis work.

Hyperspectral image (HSI) classification has been one of the research topics in the field of remote sensing. Computer vision and traditional machine learning methods have a hard time to solve the HSI classification task effectively due to the enormous amount of information in the hyperspectral image and the spectral complexity. However, deep learning, a subset of machine learning methods, managed to achieve good performance for the classification tasks [22]. Many different deep learning models, such as stacked autoencoders (SAEs) [8], deep belief networks (DBNs) [9], convolutional neural networks (CNNs) [7] and recurrent neural networks (RNNs) [24], have been used and obtained good accuracy. However, most of those studies concentrated on small hyperspectral datasets, such as Indian Pines (size of 145×145 pixels, 220 spectral bands), the University of Pavia (size of 610×340 pixels, 103 spectral bands) and Salinas (size of 512×217 pixels, 204 spectral bands) [22].

In AIROBEST, a novel convolutional neural network (CNN) has been developed by using the TAIGA dataset to simultaneously predict a wide range of forest parameters [29]. The result from the proposed model is 78.32% of balanced accuracy for categorical variables and an average mean absolute error 0.052 for continuous variables scaled to $[0, 1]$ range. Despite the decent accuracy, the test set used for the accuracy calculation is quite small and scattered and it does not reflect the result of the prediction of the whole hyperspectral image. Moreover, the result is detached from the physical measures, which makes it more difficult to evaluate it from the reality's standpoint.

Therefore, the goal of this thesis is to review and assess the predictions of the developed model from practical perspectives. Besides, some parameters of the model are also adjusted to review the changes to the results.

1.1 Problem statement

In the previous phase of AIROBEST, a multi-task learning CNN model using the hyperspectral image was developed and described in Phu Pham's Master's Thesis [29]. Inspired by Chen's model [7], the proposed CNN model consists of seven convolution layers followed by a fully-connected (FC) layer shared between the learning tasks, and two task-specific FC layers. To evaluate the performance of the model, a few metrics were introduced in the thesis, such as mean squared error and mean absolute error for the regression tasks and accuracy, precision and recall for the classification tasks. However, the

data was used to train the model in a pixel-wise manner while the original forest data is collected and presented by thousands of forest stands. Hence, the proposed metrics reflect the pixel-level prediction result from the test set only. Instead, we need new metrics and a different approach to evaluate the prediction result for the stand-level prediction.

Therefore, in the scope of this thesis, we continue using the existing model to answer the following research questions:

1. What is the difference between results acquired with the original randomized pixel-level data split and the stand-level data split? What is the best metric to compare the results between the above approaches?

2. How to quantify the variation and reliability of the predictions from the hyperspectral image?

3. What is the difference between the predictions obtained for the training and testing sets? How much is the model overfitting to the training data?

1.2 Structure of the thesis

The thesis consists of six chapters. Chapter 2 brings an overview of the hyperspectral imaging and some deep learning concepts. Chapter 3 introduces the hyperspectral forest data used in this thesis. In Chapter 4, the CNN model and the proposed metrics to evaluate the performance of the model are discussed in depth. Chapter 5 describes the experiments to answer the research questions and analyzes the results. Lastly, Chapter 6 contains the conclusion and directions for future works.

Chapter 2

Background

This chapter reviews the literature and techniques related to the topic of the thesis work. Firstly, techniques of hyperspectral imaging are introduced to understand the hyperspectral forest data. Next, the literature of deep learning, feedforward neural network, convolutional neural network and their concepts are explained. Finally, an overview of deep learning for hyperspectral images is presented.

2.1 Hyperspectral imaging

Hyperspectral imaging is an important technique in remote sensing to capture a wide wavelength range of the electromagnetic spectrum reflected by an object or an area [5]. A typical camera image has only three spectral channels in each pixel (red, green and blue), while each pixel in a hyperspectral image is a high-dimensional vector that records hundreds of spectral channels. The raw hyperspectral data is visualized as a data block. It can be considered as a stack of tens to hundreds of images with each image corresponding to a different color or a wavelength in the spectrum.

In contrast to the human brain, which only acquires three primary colors with the naked eye, a computer vision system can use more color channels. With spectral information, spectroscopy systems often have a drastic color improvement compared to the conventional tristimulus color system. In addition, hyperspectral imaging can use the near-infrared wavelength ranges which allows the system to exploit the reflections that cannot be seen by humans.

Hyperspectral imaging has been applied widely in many fields from mining, geology, surveillance, ecology and forestry [5]. In forestry and natural resource management, the technique allows to capture a large amount of in-

formation from the forest observable surfaces, thereby studying its spectral properties caused by various forest variables such as the main tree species, soil types, leaf area index (LAI) and others as will be described in Chapter 3.

2.2 Deep learning

Deep learning is a subclass of machine learning algorithms that is inspired by human brain's network of neurons to learn and perform complicated tasks such as visual recognition or fraud detection. Deep learning models are based on artificial neural networks with multiple layers to extract high-level features by transforming the raw data into more abstract representations [12].

2.2.1 Feedforward neural networks

A feedforward neural network, also called multilayer perceptron (MLPs) [32], is the most basic type of deep learning model [35]. The purpose of feedforward neural network is to find some functions f with parameters θ where $y = f(x, \theta)$ that maps the input x to its corresponding output y. By feeding the data into the feedforward network, it learns the best value of the parameter θ so that $f(x, \theta)$ is close to the true function $f^*(x)$ [12].

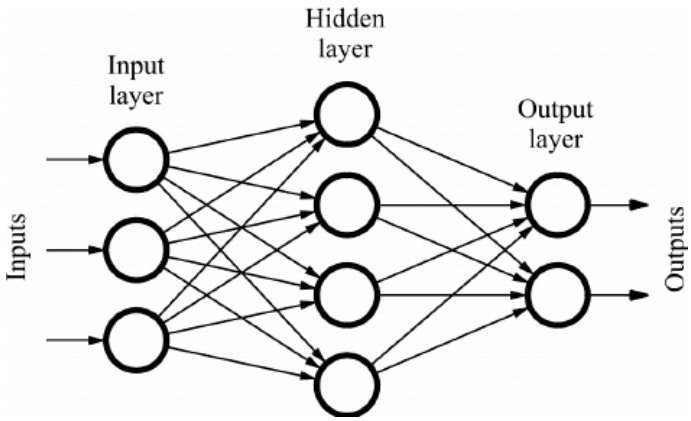


Figure 2.1: A feedforward neural network model [30].

A typical feedforward neural network has an input layer, one or more hidden layers and an output layer. Figure 2.1 is an example of a feedforward neural network with one hidden layer with four neurons. The network is called "feedforward" because the information flows from the input into the network and moves in a forward direction without feedback connections or

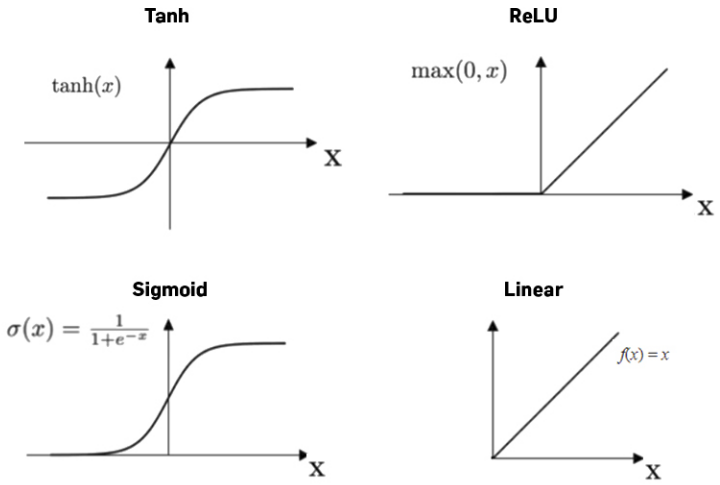


Figure 2.2: Linear activation function and three commonly used non-linear activation functions[2].

loops. Feedforward neural networks are the foundation for the more advanced networks such as convolutional neural networks, applied widely in computer vision [18, 20], and recurrent neural networks in natural language processing [15, 39].

2.2.2 Activation functions

An activation function is a function that produces an output based on an input or a set of inputs. It can be divided into two different types: linear activation function and non-linear activation functions. A linear activation function is barely used in deep learning. It transforms a neural networks with multiple layers into one layer because the last layer turns into a linear function to the first layer and the neural network becomes a linear regression model and that prevents the network to learn complex functional mappings [28].

On the other hand, non-linear activation functions are widely applied in the modern neural networks. They enables the non-linearity to the networks, which supports the learning of high order polynomials beyond one degree for deeper networks or other complicated mappings [28]. There are a few commonly used activation functions: Sigmoid, Hyperbolic Tangent (Tanh) and Rectified Linear Unit (ReLU). The graphs and formulas of these functions are shown in Figure 2.2.

Proposed by Nair and Hinton in 2010 [27], ReLU activation function has been the most popular activation function for deep learning applications with outstanding results. ReLU is nearly linear, therefore, it inherits some characteristics of linear function that make it easy to optimize with gradient-based methods [12]. In addition, it provides better generalization and performance than the Sigmoid and tanh activation functions [11, 40].

2.2.3 Loss functions

In machine learning, a loss function is the function that measures the distance score between the predicted and target values. The distance score is high if the predicted and target values are deviated much from each other and vice versa. The distance score is the way to evaluate the performance of the model and that score is used as a feedback signal to adjust the model parameters little by little to minimize the distance score on the future predictions. The adjustment is the job of the optimizer and the adjustment step is called *learning* [10].

Loss function can be categorized into two groups: ones for classification tasks and the others for regression tasks. There are a few common loss functions for classification tasks, such as Cross-entropy (CE) loss, Hinge loss, Exponential loss or Kullback Leibler divergence loss. The common loss functions for regression tasks are mean squared error (MSE) loss, mean absolute error (MAE) loss and Huber loss. In our CNN architecture, MSE and CE are selected as the two loss functions for the classification and regression tasks, respectively.

2.2.3.1 Mean squared error

Mean squared error (MSE) is the simplest and most commonly used loss function for regression. Its objective is to minimize the sum of squared difference between the predicted and target values. MSE loss function is defined as follows:

$$\begin{aligned} J_{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \ , \\ \text{where } N &= \text{the number of observations,} \\ \hat{y}_i &= \text{the predicted value,} \\ y_i &= \text{the target value.} \end{aligned} \tag{2.1}$$

[2] https://docs.paperspace.com/machine-learning/wiki/activation-function

The advantage of using MSE loss function is to reduce the outlier predictions with large errors because MSE magnifies these errors with larger weight by the squaring part of the function.

2.2.3.2 Cross-entropy loss

Cross-entropy (CE), also known as logs loss, is widely used loss function to optimize binary classification models. It is defined as follows:

$$\begin{aligned} J_{CE} &= -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \ , \\ \text{where } N &= \text{the number of observations,} \\ \hat{y}_i &= \text{the predicted value; } \hat{y}_i \in [0,1], \\ y_i &= \text{the target value; } y_i \in \{0,1\}. \end{aligned} \tag{2.2}$$

Binary CE is the negative average of the log of corrected predicted probabilities. It optimizes the difference between predicted and target distributions. For multi-class classification, the multi-class CE loss is calculated by summing up the binary CE loss for each class.

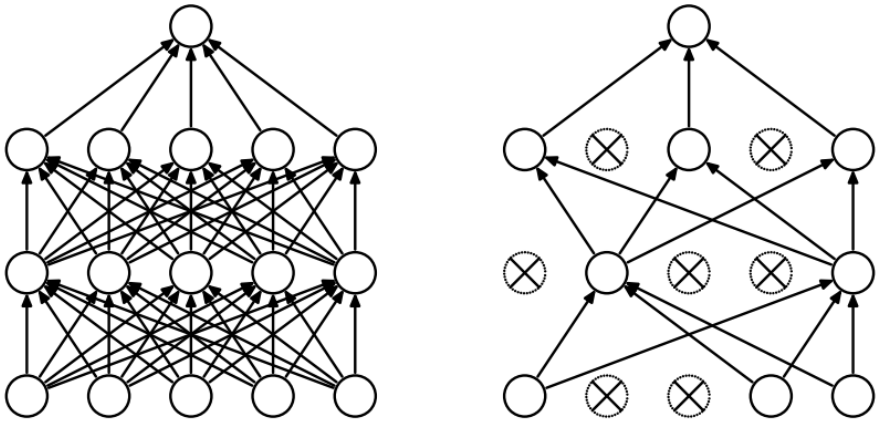
2.2.4 Regularization

One of the main issue of the model training is overfitting. It happens when a model learns the training data too detailed to the extent that it affects negatively to the test data. Regularization is a technique used to minimize the test error at the expense of increased training error to void overfitting. There are some regularization strategies used in the final model developed in [29], such as dropout and batch normalization.

2.2.4.1 Dropout

Dropout is a regularization technique that ignores or drops out some random non-output units in the network during the training process. Those units are not considered during a particular forward or backward pass. Figure 2.3 illustrates the dropout example in which the network on the left is a standard neural network and the network on the right is applied the dropout technique.

By dropping out random units, the idea of dropout technique is to train an ensemble of an exponential number of sub-networks from the original one during the training process. It performs model averaging with the neural networks. The effect is that the network becomes less sensitive to the specific



(a) Standard Neural Net (b) After applying dropout.

Figure 2.3: Dropout example [36].

weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data [36].

2.2.4.2 Batch normalization

Batch normalization is an algorithmic method of adaptive reparametrization. It helps coordinating updates across many layers in the model by normalizing the activation of each input variable (reparametrizing) to have zero mean and unit variance [12]. Subsequently, it enables training of deep neural networks faster and more stable. Optimization for convolutional neural networks benefits much from batch normalization. Algorithm 1 presents the batch normalization algorithm.

In [16], Ioffe and Szegedy defined "Internal covariate shift" phenomenon as the shifts of the distribution of the network activations due to the change in network parameters during training. Initially, batch normalization is motivated and developed by the idea of reducing that phenomenon. However, the follow-up study by Santurkar et al. suggested that might not be the case. Instead, batch normalization makes the optimization landscape smoother, thus the gradients are more predictive, and allows wider range of learning rates and faster convergence [34].

Algorithm 1 Batch normalization algorithm [16]

Input: Values of x over a mini-batch: $\beta = \{x_i \dots x_m\}$
Parameters to be learned: γ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ ▷ mini-batch mean

$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2$ ▷ mini-batch variance

$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$ ▷ normalize

$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ ▷ scale and shift

2.2.5 Convolutional neural networks

Convolutional neural network (CNN) [21] is a neural network specialized for the spatial information as the input data, such as images or time-series data. It has been applied widely to build intelligent computer vision systems with high accuracy by many big corporates, such as Facebook's 3D photo[3] or Tesla's self-driving[4].

2.2.5.1 Convolution layer

Convolution is a mathematical operation applied between a matrix and a kernel. The matrix is usually an image or a representation of an image and the kernel can be understood as a sliding window function applied to the matrix.

In Figure 2.4, the matrix on the left represents a black and white digital image. The size of the image is 7×7 and each pixel has a value of 1 or 0. The kernel, also called filter, is the blue matrix of 3×3 next to the image. Convolution is the operation of multiplying the kernel's values element-wise with the image matrix and summing them up. The kernel is sliding over the whole matrix to get the result of the full convolution. In general, the value of a neuron v_{ij}^{xy} at position (x, y) of the jth feature map on the ith layer is calculated as follows:

[3]https://ai.facebook.com/blog/powered-by-ai-turning-any-2d-photo-into-3d-using-convolutional-neural-nets/

[4]https://electrek.co/2019/11/11/tesla-train-neural-networks-self-driving-video/

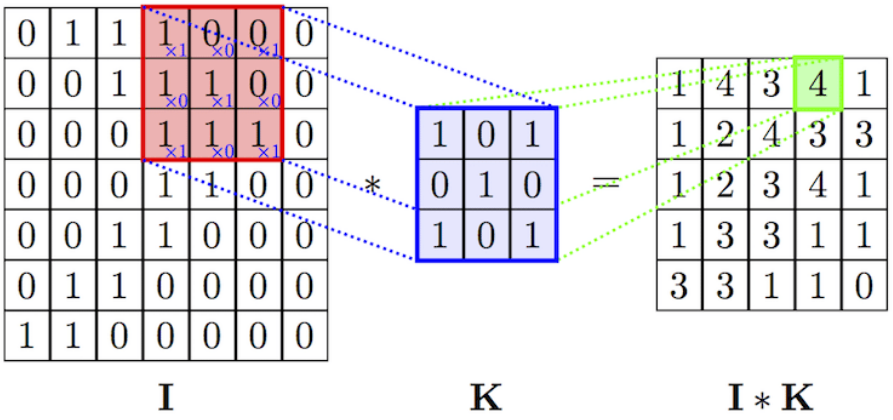


Figure 2.4: Convolution operation [33].

$$
\begin{aligned}
v_{ij}^{xy} &= g\left(\sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)} + b_{ij}\right), \\
\text{where } P_i \ \& \ Q_i &= \text{spatial convolution kernel height \& width}, \\
w_{ijm}^{pq} &= \text{the weight of position } (p,q), \\
b_{ij} &= \text{the bias.}
\end{aligned}
\tag{2.3}
$$

A convolutional neural network is a layered structure of convolution layers with non-linear activation functions and pooling layers. After passing a convolution layer and activation function, the network creates a more abstract representation of the input for the following layer. The next layer is the result of convolution from the previous layer, which leads to local connections. In addition, there are some other layers such as pooling or sub-sampling layer used to refine more useful information and eliminate noise.

During the training process, a CNN can automatically learn the values of its filters depending on the task one wants to perform. For example, in the digit recognition task, the CNN will start with raw pixels to detect the edges in the first layer, use the edges to detect the shapes in the second layer and continue to detect higher-level features such as shapes or digits in the following layers. The last layer is to classify the digits.

2.2.5.2 Pooling layer

The pooling layer is usually added between the convolutional layers for down-sampling. It reduces the spatial size of the representation to reduce the

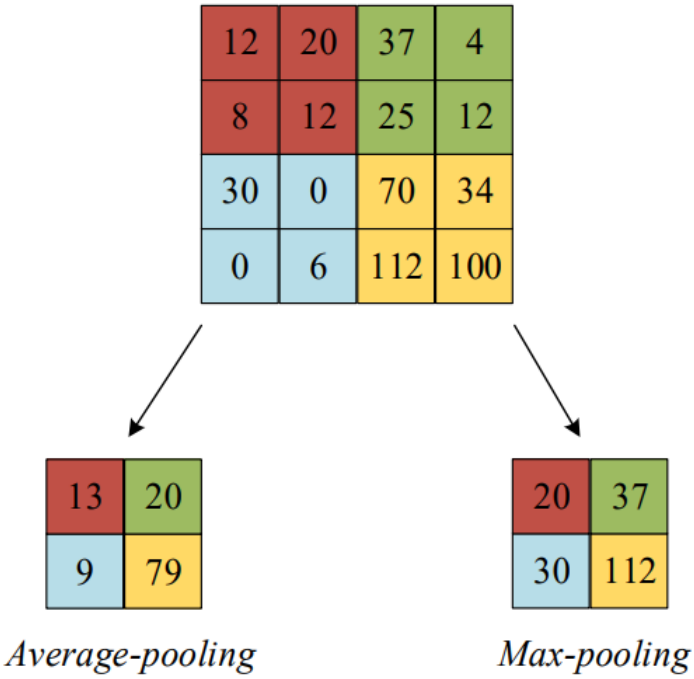


Figure 2.5: Two pooling methods with a 2×2 filter and a stride of 2 [33].

amount of parameters and computational power. In addition, the pooling layer is also helpful for extracting dominant features, which are rotational and positional invariant [12].

There are two main functions for pooling: *max pooling* and *average pooling*. Max pooling works as a noise suppressor by returning the maximum value from the part of the image covered by the filter. It performs de-noising and dimensionality reduction simultaneously. Average pooling returns an average of the values from the part of the image covered by the filter. Therefore, it only performs the dimensionality reduction. Figure 2.5 displays the pooling functions with a 2×2 filter .

2.2.5.3 Fully-connected layer

Fully-connected (FC) layer is usually added after the last convolution and pooling layers. It is also the last layer of the CNN architecture. The neurons in FC layer are connected to all neurons in the previous layer to learn the correlation between the high-level features and the output of the network. In the classification tasks, the last FC layer is normally followed by the Softmax

classifiers to output the probabilities for each class label.

2.3 Deep learning for hyperspectral images

A typical hyperspectral image is composed by hundreds of spectral bands of the same geographical location. With the detailed spectral information, the differentiating accuracy and classification accuracy of material can be improved significantly. However, with the large number of dimensions, the regular techniques to process the RGB or multi-spectral images are not as efficient. This is also known as the curse of dimensionality. Therefore, to solve the problem, the techniques such as dimensionality reduction or feature extraction are important in HSI processing [4].

In the recent years, deep learning has achieved great performance improvements and became one of the most popular techniques in the computer vision field. Therefore, deep learning is also applied to extract features of HSI. There are three different types of extracted features, including spectral, spatial and spectral-spatial features [22]. The traditional spectral-feature methods such as principle component analysis (PCA) [23], independent component analysis (ICA) [37], linear discriminant analysis (LDA) [1] are used in the early stage of HSI classification study to extract the spectral features. However, the linear transformations could not handle the spectral complexity and nonlinearity of HSI to extract deep spectral features to classify.

In [7], a 3-D CNN and its architecture are introduced to extract a joint deep spectral-spatial features from the original hyperspectral data. Figure 2.6 illustrates a 3-D convolution operation that can extract both spatial and spectral information of HSI simultaneously. The value of a neuron v_{ij}^{xyz} at position (x, y, z) of the jth feature map on the ith layer is calculated as follows:

$$
\begin{aligned}
v_{ij}^{xyz} &= g\left(\sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)} + b_{ij}\right), \\
\text{where } P_i \;\&\; Q_i &= \text{spatial convolution kernel height \& width}, \\
R_i &= \text{kernel size toward spectral dimension}, \\
w_{ijm}^{pqr} &= \text{the weight of position } (p, q, r), \\
b_{ij} &= \text{the bias}.
\end{aligned}
\tag{2.4}
$$

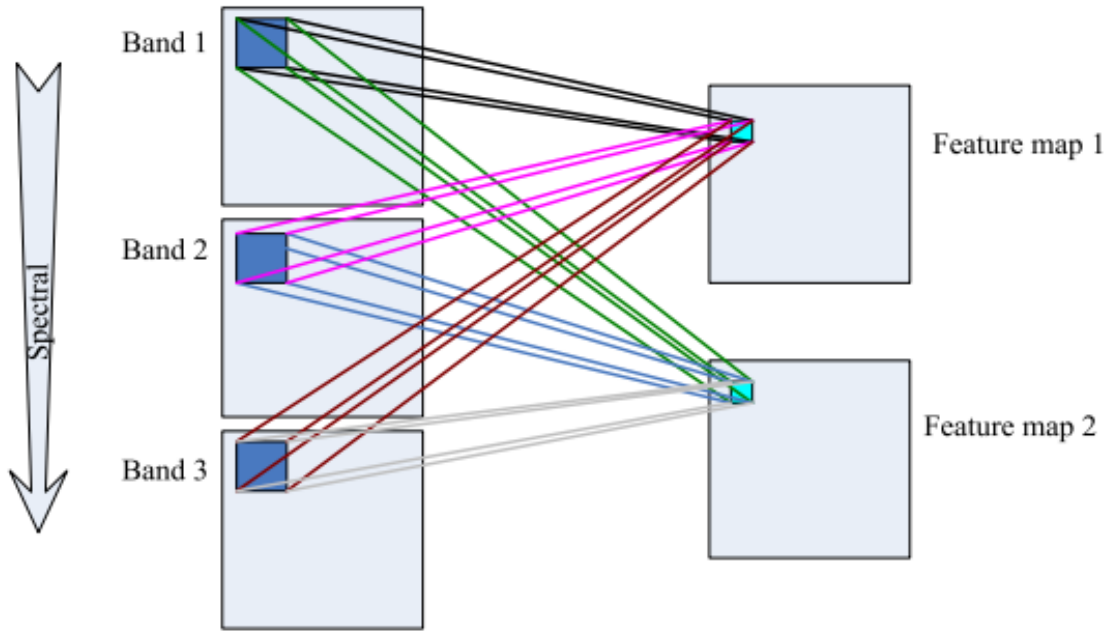


Figure 2.6: 3-D convolution operation [7].

2.4 Multi-task learning

In machine learning, one usually aims to solve a single task, whether it is classification or regression, at a time. The model is fine-tuned and optimized for a particular metric such as MAE for a continuous task or cross-entropy loss for a categorical task to a point that its performance no longer increases. However, by focusing on one single task, we might ignore a lot of information that would help the model to reach a better performance, especially for the hyperspectral image, which has an enormous amount of information. Multi-task learning (MTL), a sub-field of machine learning, tackles this problem by utilizing the similarity between the tasks to solve multiple tasks simultaneously [31].

The inspiration of MTL is to copy how a normal human learns to do a particular task. Humans usually learn to perform one task by applying the knowledge they acquire from other tasks. For example, a person who can speak Spanish is likely to take less time to learn Italian than two different people learn to speak Spanish and Italian from the beginning because both languages are derived from Latin.

From the machine learning point of view, MTL has empirically and the-

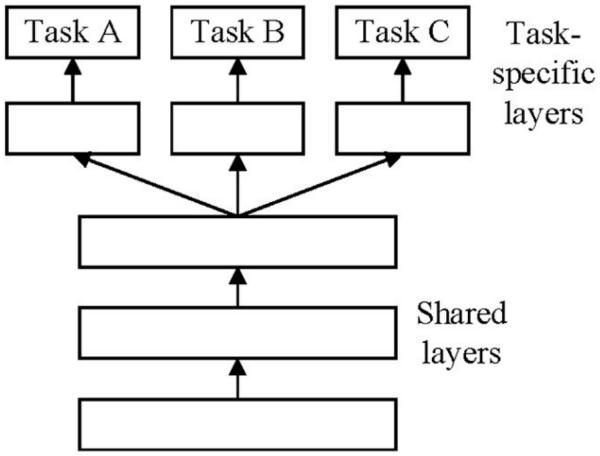


Figure 2.7: Hard parameter sharing for multi-task learning [38].

oretically proven that jointly learning multiple tasks shows a better performance than learning the tasks individually [6]. One of the benefits of MTL is that it implicitly works as a regularizer. As all tasks have different noise patterns, MTL makes the model to ignore the data-dependent noise and learns the general representation. In addition, if the data is noisy or high-dimensional, selecting the relevant features might be difficult for single-task learning. However, MTL enables the model to focus on the features that matter because the other tasks also participate in selecting the relevant features. If some features are difficult to learn for some task A but simple to learn for some task B, MTL helps the model to learn those features through task B (by eavesdropping [31]) instead. Lastly, MTL prefers the representations that most of the tasks prefer. It might help the model to expand to new tasks in the future.

Based on the nature of the tasks, there are some different settings of MTL: multi-task supervised learning, multi-task unsupervised learning, multi-task semi-supervised learning, multi-task reinforcement learning, multi-task active learning and multi-task online learning [31]. In this thesis work, a multi-task supervised learning setting is used since each task is a supervised learning task. There are two MTL approaches specific for deep learning: *hard parameter sharing* and *soft parameter sharing*.

Figure 2.7 shows the architecture of hard parameter sharing for multi-task deep learning [6]. It is the most popular approach to MTL in neural networks. The architecture contains two parts: shared layers and task-specific layers. The shared hidden layers help the model to learn a representation

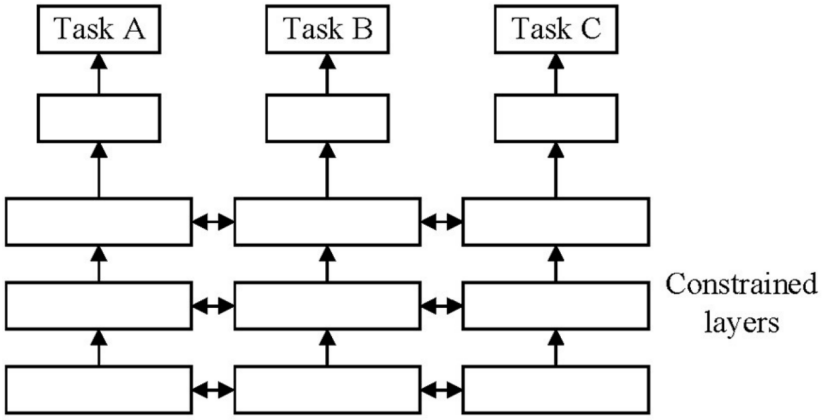


Figure 2.8: Soft parameter sharing for multi-task learning [38].

that generalizes well for all the tasks. It acts as a regularization method to reduce the risk of overfitting. MTL with hard parameter sharing is applied in state-of-the-art AI systems such as Tesla's Autopilot [5].

On the contrary, the soft parameter sharing approach requires that each task has its own model and parameters [31]. This approach also contains two parts: the constrained layers and the task-specific layers. The constrained layers regularized the distance between the parameters of the model to encourage the parameters to be similar to represent all tasks. It also gives more flexibility for the tasks than the hard parameter sharing approach by the loose links in the constrained layer. Figure 2.8 shows the architecture of soft parameter sharing for multi-task deep learning.

[5]https://slideslive.com/38917690/multitask-learning-in-the-wilderness

Chapter 3

TAIGA hyperspectral and forest data

This chapter studies the TAIGA hyperspectral remote sensing data and the reference forestry data used in the thesis work. In addition, different data split techniques are also explained.

3.1 Remote sensing data

In the project, the neighborhood of Hyytiälä forestry field station, located in Central Finland, has been chosen as our study site. The selected site consists of agriculture fields, wetlands and boreal forests, where the majority are covered by Scots pine, Norway spruce and silver birches. Most of the forest stands in the area are mixed [13].

The hyperspectral AISA (Airborne Imaging Spectrometers for Applications) was used to capture the TAIGA dataset of the Hyytiälä site under the clear sky condition. The flight campaign happened on June 15, 2017 with the altitude 980 meters above the ground level. The hyperspectral image (HSI) was acquired from nine consecutive flight lines with the average solar zenith and azimuth angles were 41.32° and 149.49°, respectively [13]. The total covered area in HSI is approximately 3000 ha (3.3 km wide and 9 km long). The image has a resolution of 12143×12826 pixels and 128 spectral bands. Figure 3.1 shows the RGB format of our hyperspectral image (HSI) from the study site.

Before using the hyperspectral data to train the model, the data needs to be pre-processed to reduce its noise content. In the pre-processing step, the first 110 bands were used while the last 18 bands with wavelengths over 910 nm were discarded.

Figure 3.1: RGB image of the TAIGA hyperspectral forest data.

3.2 Reference forest data

In forestry, a stand is a contiguous group of trees that shares the same age, tree type, height, size or spatial arrangement to differentiate with the adjacent groups. A collection of stands is a forest. The TAIGA dataset for our studies is based on the forest resource data gathered and processed in the inventory process by the Finnish Forest Centre[1].

Throughout the inventory process, the forest data are collected in nationwide field measurements and airborne remote sensing by using airborne laser scanning and aerial imaging. The forest variables are measured and inputted for an inventory unit, a 16m $\times$ 16m grid cell. The stand-level forest data are calculated by averaging the forest variables of the grid cells.

In this thesis work, the label data are provided as stand-level forest data. They contain the forest characteristics of the stands, such as fertility class, mean height or basal area. The data is then processed further to pixel-level forest data to be used in CNN model. The area of the TAIGA dataset has

[1] `https://www.metsakeskus.fi/`

1703 stands with labels provided, however, only 671 stands were used to generate the pixel-level data to train a forest variable predictor [14]. In addition, the forest stand boundaries were set to 10 meter buffering to avoid the geolocation errors and neighborhood issues. Figure 3.2 shows the hyperspectral image overlaid by the stands and their division in training and test splits in [14].

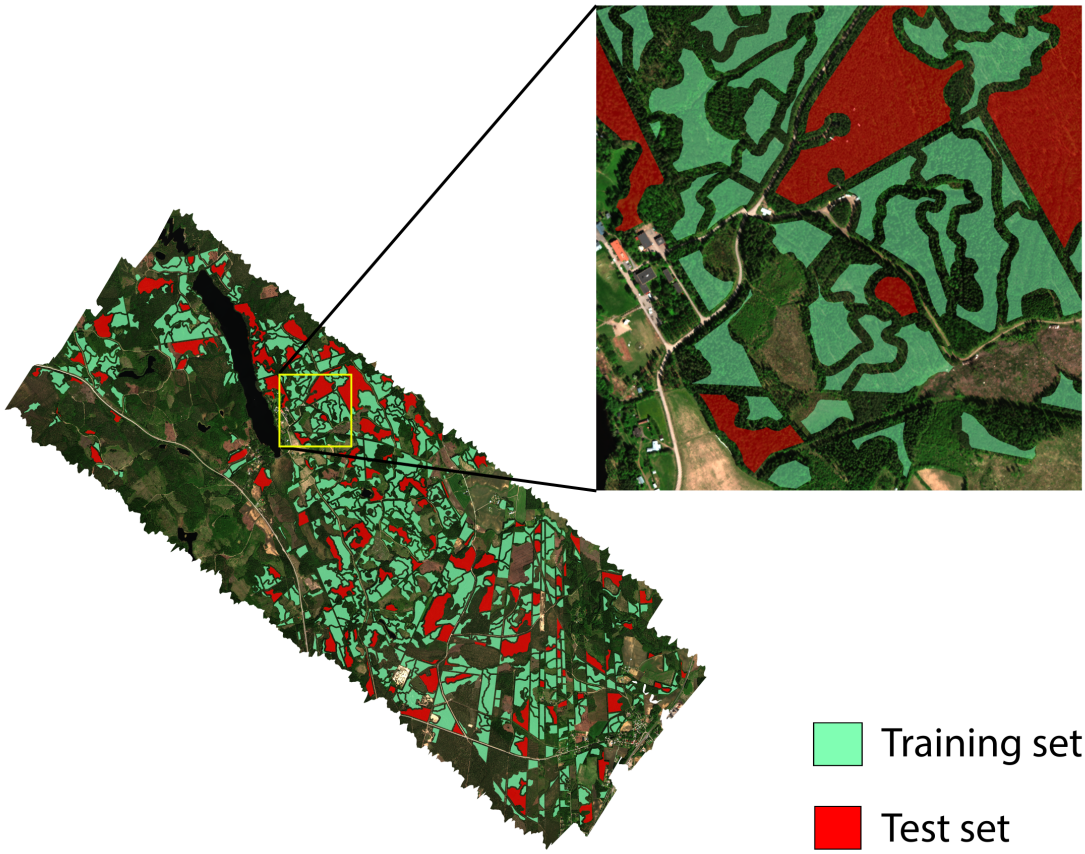


Figure 3.2: Forest stands in the used TAIGA hyperspectral forest data.

3.3 Data processing

Table 3.1 shows the number of forest variables in TAIGA. There are thirteen forest variables three of which are categorical variables and the other ten are continuous. Since some of the forest variables are directly measured while

#	**Forest variable**	**Unit**	**No. of classes or range**
Categorical			
1	Fertility class	unitless	4
2	Soil class	unitless	2
3	Main tree species	unitless	3
Continuous			
4	Basal area	m^2/ha	0 – 35.51
5	Mean DBH	cm	0 – 30.89
6	Stem density	1/ha	0 – 6240
7	Mean height	m	0 – 24.16
8	Percentage of pine	%	0% – 100%
9	Percentage of spruce	%	0% – 84%
10	Percentage of birch	%	0% – 58%
11	Woody biomass	t/ha	0 – 180
12	Leaf area index	unitless	0 – 9.66
13	Effective leaf area index	unitless	0 – 6.45

Table 3.1: Forest variables in TAIGA.

others are derived, the data is quite noisy and needs to be pre-processed before passing to the model.

Table 3.2 shows the original class distribution of the categorical variables. Each categorical variable has some major and minor classes and the minor classes only account for less than 1.0% of the samples. The class imbalance of the data is a typical problem in machine learning classification tasks and it might lead to poor performance because the minority classes are usually ignored. In order to fix this issue, we removed the classes that account for less than 5% of the variable in the pre-processing step.

All of the ten continuous variables has a different unit and value range from each other, therefore, before feeding to the model, each continuous variable is normalized to the $[0, 1]$ range by using min-max scaling. However, some of the variables have some extremely high values in a few pixels, which might push the majority of the values toward zero when normalizing. To solve this issue, the label values within 98^{th} percentile of the min-max values were hold to keep the majority of the sample values and the rest were clipped in the normalization. Figure 3.3 illustrates the normalized labels of basal area.

#	Categorical variable	Class Name	Distribution
1	Fertility class	**Mesic heath forest**	**62.06%**
		Sub-xeric heath forest	**23.45%**
		Xeric heath forest	**8.71%**
		Herb-rich heath forest	**5.05%**
		Barren heath forest	0.46%
		Herb-rich forest	0.26%
2	Soil class	**Mineral**	**86.44%**
		Organic	**13.56%**
2	Main tree species class	**Scots pine**	**60.29%**
		Norway spruce	**33.05%**
		Birch and other broadleaves	**6.66%**

Table 3.2: Original class distributions of the categorical variables.

Set	**Number of samples**	
	w/o augmentation	**w/ augmentation**
Training	17594	42832
Validation	1955	4759
Test	4888	4888

Table 3.3: Number of data samples of train, validation and test sets with a patch size of 27×27 pixels using the non-overlapping data split.

3.4 Data split

The CNN model requires a lot of image samples to train and, therefore, it is important to have a large training set. The samples were extracted from TAIGA by splitting the hyperspectral image into small patches.

In the previous study of AIROBEST, the hyperspectral image was split into non-overlapping patches of size 27×27 pixels [29]. The patches that contain non-forest area, such as roads, lakes, etc, were dropped from the dataset due to lack of label data. The selected patches were then randomly shuffled and divided into training (72%), validation (8%) and test sets (20%). Table 3.3 shows the number of samples extracted by the pixel-wise non-overlapping data split. If the patch size is increased, the number of samples decreases and vice versa because of the non-overlapping requirement. Different patch sizes were tested and 27×27 pixel patches gave the best overall results for both categorical and continuous variables.

In another study, Halme et al.[14] experimented with stand-level data for both training and test sets for the two machine learning regression algorithms:

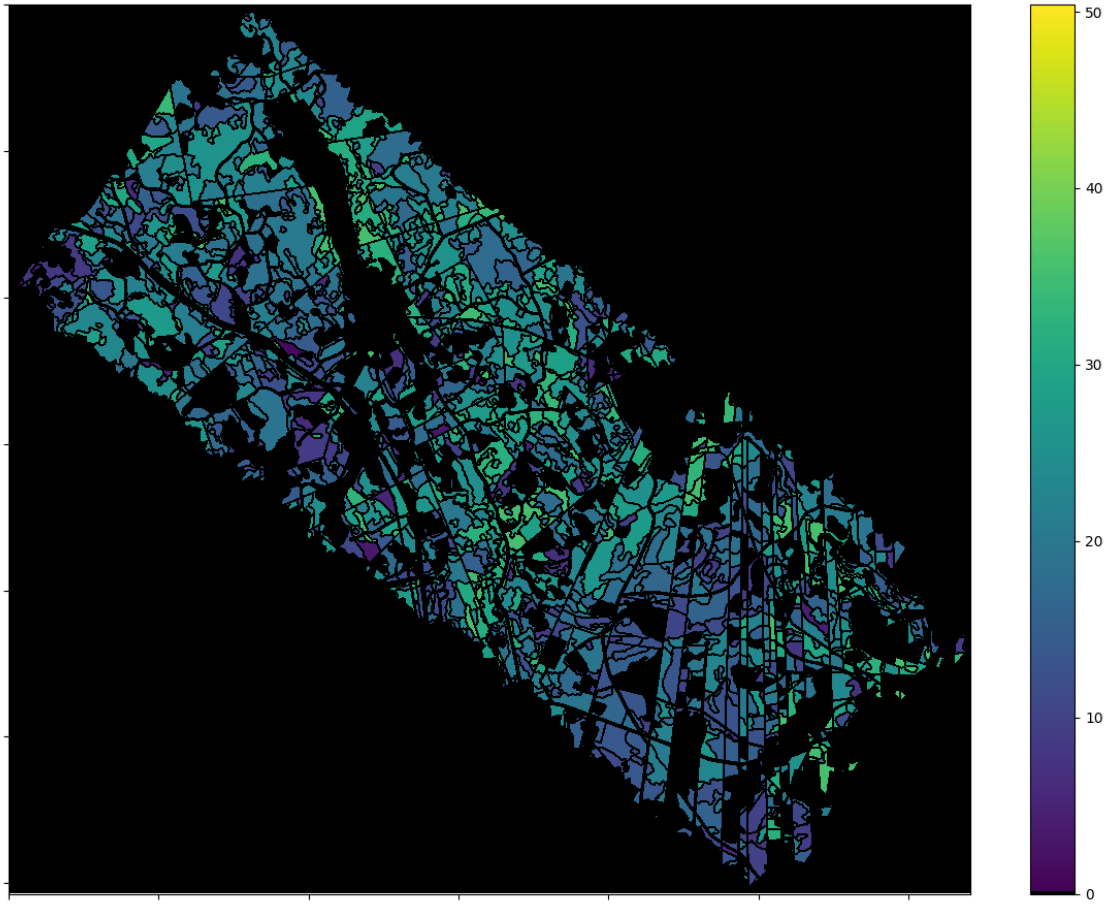


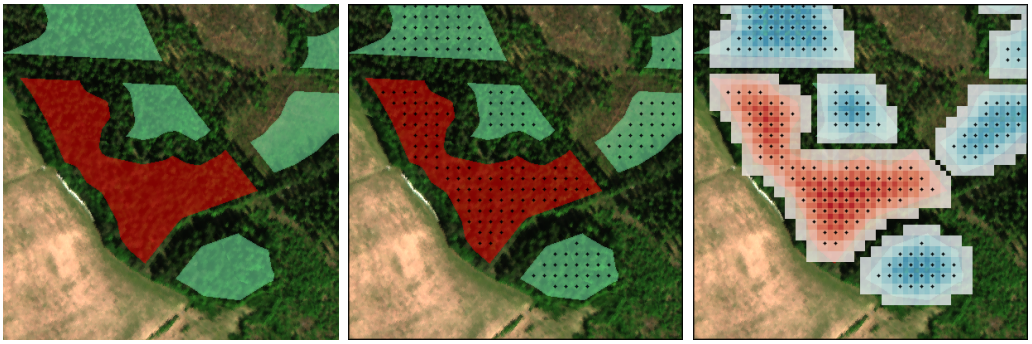
Figure 3.3: Continuous values of basal area in m^2/ha.

Gaussian process regression (GPR) and support vector regression (SVR). Figure 3.2 illustrates the geographical location of the selected stands.The stand-level data cannot be fed into the CNN model directly and therefore, in this study it is used to generate the pixel-level data. 671 stands are selected for this study, of which 560 stands (83%) and in the training and validation set and 111 stands (17%) in the test set.

The samples are extracted from the stands by splitting each stand into smaller patches. The central pixels of the patches are picked from the stands with different stride sizes. The training and test sets are split from the stands to reduce the risk of overlapping when the patch size is increased significantly, which might lead to the overfitting problem. The stand-level test set is extracted into the pixel-level test set while the stand-level training

Model family		S1	S2	S3
Stride size		17×17	13×13	11×11
Set	**Stands**	**Number of samples**		
Training	⟩ 560 ⟨	19702	37749	55612
Validation		2190	4195	6180
Test	111	10949	20263	20263

Table 3.4: The numbers of samples of train, validation and test sets with different stride size of patch sampling.



(a) Stands (b) Central Pixels (c) Samples

Figure 3.4: Stand-level data split.

set is extracted into both the pixel-level training set (90%) and validation set (10%).

Table 3.4 shows the number of samples extracted when using different stride sizes. As the stride size decreases, the number of samples increases considerably. The results comparison between the stride size is reported in Chapter 5. Having too many samples slows down the model training significantly and therefore, the stride size of 13×13 pixels is selected in this study because the number of samples is then also quite similar to the previous study [29].

Figure 3.4 displays the steps to extract the samples from the stands. In the last step, the patches with patch size of 45×45 are used to show the area of the pixel-level training and test sets. As the patch size is large enough, some pixels might appear in both training and test sets, hence, they are overlapping. Figure 3.5 illustrates the overlap percentage between the training and test sets as the patch size increases. As the patch size of 45 ×45 pixels, the overlap percentage is 0.60% which is very small compared to other the popular hyperspectral data sets [25].

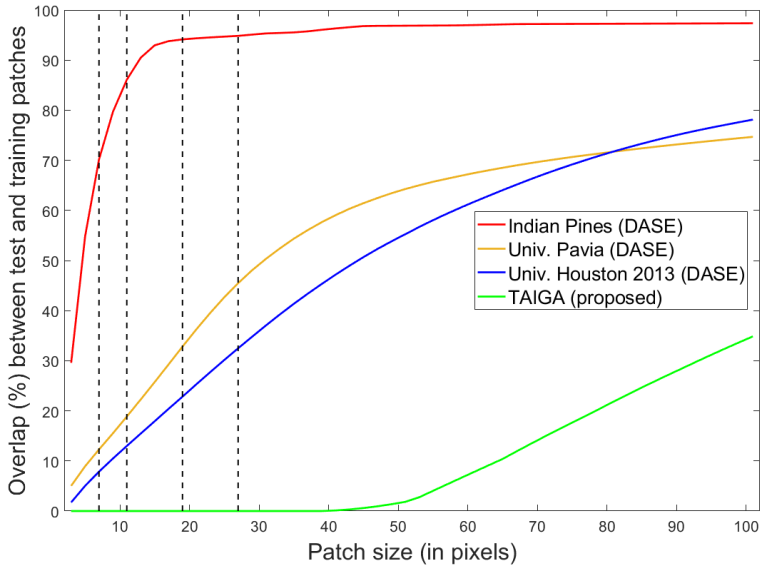


Figure 3.5: Overlap percentage between training and test patches as a function of patch size, for different hyperspectral datasets [3] [2, 19, 25] .

[3] http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

Chapter 4

Methods

In this chapter, the CNN architecture from the previous phase of AIROBEST is reviewed in details. In addition, a new set of metrics are proposed to evaluate the accuracy of the stand-wise forest variable prediction.

4.1 CNN architecture

Figure 4.1 shows the CNN architecture for HSI multi-task learning that is utilized in Phu Pham's MSc Thesis work [29]. Inspired by spectral-spatial feature extraction 3-D CNN by [7], the original CNN architecture proposed in [29] consists of seven convolution layers for feature extractions and three fully-connected (FC) layers for multi-task learning of the forest variables. In this thesis, only the best performing CNN model is described.

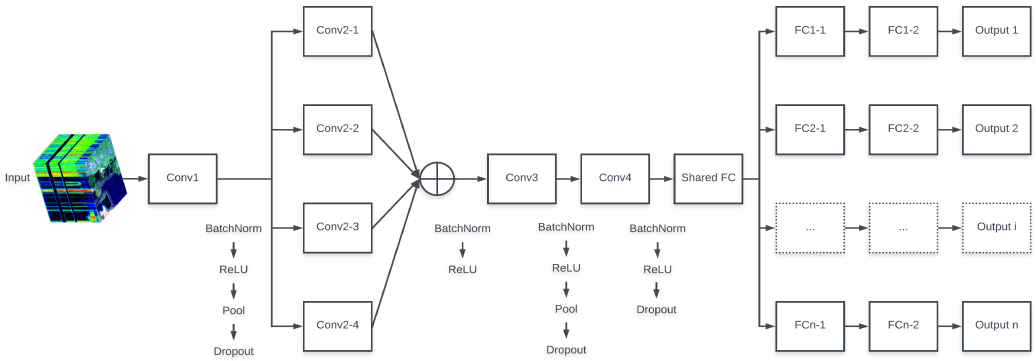


Figure 4.1: Original CNN architecture for HSI multi-task learning [29].

As shown in Figure 4.2, the first layer has one 3-D convolution (Conv1)

of 3×3×54 kernel with a stride of one, padding of zero and an output of 128 channels. The output of the convolution is followed by batch normalization that helps the model train faster and more stable. The batch normalization makes the model less sensitive to the learning rate and parameter initialization. After the batch normalization, the output is applied to a non-linear ReLU activation function followed by a pooling operation with a kernel of size 2×2×1. Finally, dropout is used as the regularization technique to avoid the risk of overfitting.

Following the first 3-D CNN layer is a multi-scale convolution block, which consists of four convolution operations named from Conv2_1 to Conv2_4 with the kernel size of 1×1×1, 1×1×3, 1×1×5, 1×1×11, respectively. Correspondingly, the padding of the convolutions are 0×0×0, 1×0×0, 2×0×0 and 5×0×0. The outputs of the convolution operations are summed together before going through a batch normalization and a ReLU function. The purpose of the multi-scale convolution block is to extract and exploit the spectral correlations as the kernel is single dimensional.

In the second and third layers (Conv3 & Conv4), the convolution operations have a kernel of size 3×3×32 with a tride of one and padding of zero. The outputs of Conv2 and Conv3 are 64 and 32 channels, respectively. Both the outputs are fed to a batch normalization, ReLU function and dropout in the end. The pooling operation is only applied to the second layer with a kernel of size 2×2×1. The output of the third layer is flattened and fed to the multi-task learning block.

In this architecture, multi-task learning (MTL) uses the hard parameter sharing approach [29]. It contains one shared fully-connected (FC) and two non-shared FC layers. The shared FC takes the output of the third 3-D CNN layer and transforms it into a 512-length feature vector. The two non-shared FC layers are used to predict the individual forest variables. The 512-length feature vector is fed into the first non-shared FC and transformed into a 300-length feature vector. If the task is a continuous task, the second non-shared FC is used to predict the continuous value directly. Otherwise, if the task is a categorical task with N classes, the second non-shared FC converts 300-length feature vector into a N-length vector and uses the Softmax function to calculate the probabilities for each class.

For the loss function, mean square error (MSE) is used for the continuous tasks and cross entropy is used for the categorical tasks. The model is trained with the learning rate of 10^{-4}, batch size of 32 samples with different patch sizes. The Adam optimizer is used with the weight decay at 10^{-4}.

The CNN model can predict the labels for a pixel in HSI by using a patch of the surrounding pixels as an input data with a size of K×K×B in which B is 110 bands of the hyperspectral image and K is the patch size. In the

Layers	Conv1	Conv2-1	Conv2-2	Conv2-3	Conv2-4	Conv3	Conv4
kernel size	$3 \times 3 \times 54$	$1 \times 1 \times 1$	$1 \times 1 \times 3$	$1 \times 1 \times 5$	$1 \times 1 \times 11$	$3 \times 3 \times 32$	$3 \times 3 \times 32$
padding	0	0	$0 \times 0 \times 1$	$0 \times 0 \times 2$	$0 \times 0 \times 5$	0	0
out channels	128	128	128	128	128	64	32
pooling	$2 \times 2 \times 1$	–	–	–	–	$2 \times 2 \times 1$	–

Layers	Shared FC	FCn-1	FCn-2
in channels	varied	512	300
out channels	512	300	N or 1

Figure 4.2: Configurations of the layers.

scope of this thesis, different patch sizes and different methods to extract the pixel are used to train the CNN models, and the models are evaluated against each other in Chapter 5.

4.2 Evaluation metrics

A set of evaluation metrics were discussed and applied in Phu Pham's MSc Thesis work [29]. However, those metrics were more suitable for evaluating the pixel-level predictions than the stand-level ones. Therefore, in this section, several new metrics for both categorical and continuous tasks are introduced to evaluate the stand-level forest variable predictions.

4.2.1 Root mean square error

Root mean square error (RMSE) is one of the most frequently used evaluation metrics for the regression variables. It is defined as

$$\begin{aligned} RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \ , \\ \text{where } N = \text{the number of observations,} \\ \hat{y}_i = \text{the predicted value,} \\ y_i = \text{the target value.} \end{aligned} \tag{4.1}$$

RMSE is the square root of mean square error (MSE) and can be interpreted as the standard deviation of the prediction errors or the difference between the predicted and target values. A small RMSE indicates that the model is fit well. RMSE has the same unit as the forest variables being measured so it is easily comparable with the target value. The stand-level

RMSE is calculated similarly and N is the number of stands, $\hat{y}_i$ is the stand-level predicted value and y_i is the stand-level target value. The stand-level predicted value is calculated by averaging the pixel-level predicted values for each test stand.

4.2.2 Relative root mean square error

In the model, there are ten different regression variables and each variable has different units. Therefore, RMSE is not suitable to compare the prediction performance between the variables. Relative root mean squared error (rRMSE) is a good alternative evaluation metric as it is presented as percentages:

$$rRMSE = \frac{RMSE}{\bar{y}} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}}{\frac{1}{N}\sum_{i=1}^{N} y_i} \quad . \tag{4.2}$$

rRMSE is the RMSE normalized with the mean of the target values. The smaller the rRMSE, the better the model is. The stand-level rRMSE equation is the same with the stand-level predicted and stand-level target values.

4.2.3 Relative bias

The bias characterizes the systematic difference between the predicted and target values. A forest variable with high bias value means that the model is too simple for that variable and underfitting. Relative bias, like rRMSE, is an alternative metric to compare the bias between the forest variables. It is defined as:

$$rBias = \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)}{\sum_{i=1}^{N} y_i} \quad . \tag{4.3}$$

Relative bias is calculated as the sum of the bias values divided by the sum of the target values. The model is the better the closer relative bias is to 0%.

4.2.4 Coefficient of determination (R^2)

The coefficient of determination (R^2) represents the goodness of fit and how well unseen samples are likely to be predicted by the model [26]. R^2 is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\bar{y} - y_i)^2} \ , \quad \text{where } \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \ . \tag{4.4}$$

R^2 equals to the squared Pearson correlation coefficient of the predicted and target values. The closer the R^2 score is to 1.0, the better fit the model is.

4.2.5 Standard deviation

Standard deviation is the measure of the variation of a sample set. A low standard deviation shows the values of the set are closer to its mean while a high standard deviation shows the values are more spreading out over a wider range. It is defined as:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N}} \quad \text{where } N = \text{sample size}, \ \bar{y} = \text{sample mean}. \tag{4.5}$$

The standard deviation of a forest stand measures the variance of the pixel-level predicted values of that stand. Hence, the low stand-level standard deviation indicates the predicted values of the pixels are close to each other. A 5×5 pixels standard deviation measures the variance of the surrounding pixels to the selected pixel. The high 5×5 pixels standard deviation indicates the predicted values of the surrounding pixels are not same as the value of the selected pixels.

4.2.6 Confidence interval

Confidence internal (CI) is the estimated range of values calculated from a given set of sample data. It gives the range of values for an unknown parameter, such as a forest variable, based on the confidence level. The confidence level of an interval gives the probability that the true value of the parameter is included in the interval. The greater the confidence level, the wider the confidence interval. Common selections for the confidence level are 90% and 95%. Figure 4.3 illustrates the 90% and 95% confidence interval.

To get a 90% confidence interval, 5% of the bottom and the top of the sorted sample set are excluded. Similarly, by ignoring 2.5% of the bottom

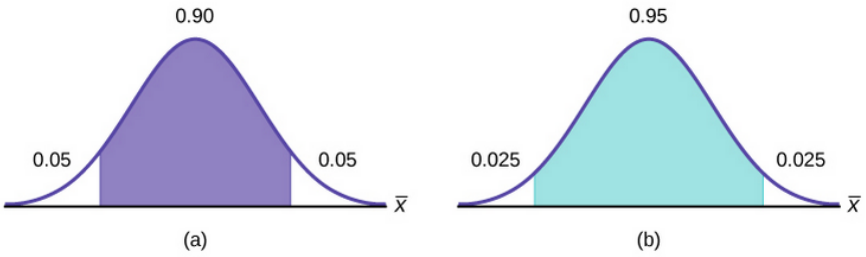


Figure 4.3: 90% and 95% confidence intervals[1].

and the top of the sorted sample set, we get the 95% confidence interval. The width of the confidence interval is then calculated by differentiating the upper bounce and lower bounce. The width of the interval increases as the confidence level increases or the sample size decreases. The width's unit is same as that of the forest variable in question.

[1] https://courses.lumenlearning.com/introstats1/chapter/a-single-population-mean-using-the-normal-distribution/

Chapter 5

Experiments and results

In this chapter, the experiments and results of the proposed CNN architecture are presented. Many models with different configurations are trained and evaluated thoroughly using the metrics from Chapter 4. The best model is selected to be used further in the subsequent experiments. Each subsection aims to answer one of the research questions in the Introduction chapter.

5.1 Pixel and stand-level split

The first experiment aimed to answer the research question about the difference between results acquired with the original randomized pixel-level data split and the stand-level data split. We are using the original randomized pixel-level data split and the new stand-level data split described in Section 3.4 to train the CNN model with the best performing configuration, detailed in Section 4.1. The patch size of 27×27 pixels is used in the first experiment. For the stand-level data split, we are using the stride size of 17×17 pixels to create the training, validation and test sets. The original randomized pixel-level data model is named P1 and the new stand-level data model is named S1-27. The models are trained for 150 epochs and evaluated with the proposed metrics. We are also using the result from [14] to compare with our results.

In Halme's study [14], Support vector regression (SVR) and Gaussian process regression (GPR) are selected as the regression algorithms to predict the forest variables. The models are trained with the stand-level data instead of the pixel-level data mentioned above. There are four different forest variables, *mean height*, *basal area*, *effective LAI* and *woody biomass*, and the models are trained separately for each forest variable.

Table 5.1 shows the evaluation results from the four above models. Out

Forest variable	SVR	GPR	P1	S1-27
RMSE				
mean height (m)	2.88	2.60	3.82	**2.30**
basal area (m^2/ha)	3.84	3.85	3.94	**3.83**
effective LAI	0.72	**0.66**	1.18	0.79
woody biomass (t/ha)	22.50	21.78	26.42	**23.06**
rRMSE				
mean height	17%	15%	22%	**13%**
basal area	17%	17%	20%	**17%**
effective LAI	22%	**20%**	27%	24%
woody biomass	29%	28%	28%	**24%**
Relative bias				
mean height	–3%	**–2%**	–16%	–4%
basal area	–2%	**0%**	–11%	–2%
effective LAI	–2%	**0%**	–8%	–4%
woody biomass	–6%	**–3%**	–15%	–5%
$\boldsymbol{R^2}$				
mean height	0.59	0.66	0.34	**0.70**
basal area	0.68	0.67	0.43	**0.68**
effective LAI	0.81	**0.83**	0.63	0.74
woody biomass	0.61	0.62	0.57	**0.68**

Table 5.1: Stand-level evaluation of the different models. SVR and GPR results are from [14].

of all the models, P1 has the worst result in all the metrics. S1-27 has the best average rRMSE and coefficient of determination, but as the individual forest task, GPR and SVR perform better in effective LAI. On the contrary, both SVR and GPR are better than P1 and S1-27 in the relative bias metric. This is probably because each forest variable has its own model for both SVR and GPR while P1 and S1-27 are multi-task learning models. In general, the results suggest that with the same CNN model, the new stand-level data split and the averaging of the predictions provide better predictions than the original one, hence it leads better result for the stand-level evaluation metrics. The results also show that the 3-D CNN architecture provides better model than the machine learning regression algorithms as the average rRMSE is lower even though the same model is used to predict all the forest variables instead of specific model for each variable.

Table 5.2 shows the evaluation results of all forest variables of the model S1-27. The categorical forest variables achieved a mean overall accuracy of

76% with the mean class accuracy of 74%. The ten continuous variables also achieved a mean rRMSE of 26.25%, mean relative bias of –1.82% and 0.74 for the coefficient of determination. Mean height is the most accurate continuous forest variable with only 13.19% of rRMSE while percentage of birch is the worst variable with 51.72% of rRMSE. From here onward, the four main continuous variables shown in Table 5.1 and bolded in Table 5.2 will be considered together averaged as shown in the latter table.

Variable #	**Variable name**	**Evaluation metrics**					
Categorical		overall accuracy (micro)			mean class accuracy (macro)		
1	Fertility class	57.66 %			61.34 %		
2	Soil class	83.78 %			80.81 %		
3	Main tree species	86.49 %			82.98 %		
mean		**75.98** %			**75.04** %		
Continuous		RMSE	rRMSE	rBias	R^2	90%CI	95%CI
4	**Basal area [m^2/ha]**	3.83	16.98%	–2.18%	0.68	13.28	14.02
5	Mean DBH [cm]	3.07	14.36%	–4.15%	0.70	9.19	13.52
6	Stem density [1/ha]	372.62	39.94%	5.57%	0.74	600.99	1097.86
7	**Mean height [m]**	2.30	13.19%	–3.72%	0.70	7.03	9.73
8	Percentage of pine [%]	12.04	24.11%	1.26%	0.85	35.85	40.49
9	Percentage of spruce [%]	10.96	30.12%	–2.87%	0.79	32.51	38.94
10	Percentage of birch [%]	6.36	51.72%	0.21%	0.76	23.46	31.51
11	**Woody biomass [t/ha]**	23.06	23.67%	–4.66%	0.68	76.16	87.45
12	Leaf area index	1.17	23.98%	–3.67%	0.73	3.80	4.71
13	**Effective leaf area index**	0.79	24.43%	–3.98%	0.74	2.63	3.14
mean*		—	**19.46%**	**–3.56%**	**0.69**	—	—

Table 5.2: Evaluation result of all forest variables for the best model S1-27. The mean of the continuous variables is for the four bolded ones.

We continue the experiment by exploring the variants of the model to find the best patch size and stride configuration. The CNN architecture uses the multi-task supervised learning settings in which each task is a supervised learning task. The model needs to predict 13 forest variables simultaneously including three categorical and ten continuous variables. Therefore, a variant of the model S1-27 that only predicts the continuous variables, is trained to compare. Our findings in Table 5.3 show that the new variant model has clearly worse results in every metrics. The mean rRMSE is 24.45% comparing to 19.46% of S1-27. This suggests that the categorical variables are correlated with the continuous ones and help predicting them better in the multi-task learning setup.

Table 5.3 also indicates the correlation between the number of training samples and the mean rRMSE. The variant S1-27 w/aug of S1-27 uses a training set that is augmented to double the training samples. Image flipping is the applied of the augmentation method. The other two variants S2-27 and S3-27 use different stride sizes mentioned in Section 3.4. S2 and S3 imply

Model	stride size	training samples	mean rRMSE	mean rBias	mean R^2
S1-27	17×17	19702	19.46%	–3.56%	0.69
S1-27 only cont.	17×17	19702	24.46%	–15.02%	0.48
S1-27 w/aug	17×17	29404	18.92%	–0.96%	0.71
S2-27	15×15	37749	18.34%	–1.62%	0.72
S3-27	13×13	55612	**17.73%**	**–0.84%**	**0.74**

Table 5.3: Evaluation results of variants of model S1-27.

the model families that use the stride size of 15×15 and 13×13, respectively. The stride size can be as small as 1×1 pixel to use all the pixels in the training stands. However, increasing the number of training samples also means it requires more training time and possibly also increasing the risk of overfitting, so a handful of methods are used to create the samples. The number of training samples in S2-27 and S3-27 are almost doubled and tripled comparing to S1-27, but mean rRMSE decreases slightly by 0.58% and 1.19 %, respectively. It can be concluded that as the number of training samples increases, the model has lower mean rRMSE and higher mean coefficient of determination and, hence, the model is better fit. However, the metrics are not improved proportionally to the number of training samples.

Model	patch size	overlap %	mean rRMSE	mean rBias	mean R^2
S2-27	27×27	0%	18.34%	–1.62%	0.72
S2-33	33×33	0%	18.27%	–2.49%	0.75
S2-39	39×39	0%	17.50%	–2.36%	0.75
S2-45	45×45	0.60%	**17.24%**	**0.30%**	**0.75**
S2-75	75×75	17.64%	17.73%	–1.80%	0.73
S2-91	91×91	28.58%	18.65%	–2.66%	0.72

Table 5.4: Evaluation results of the different patch size models.

The last experiment to find the best configuration is exploring the different patch sizes. A patch size of 27×27 means that the information of 27×27 pixels is used to predict the central pixel. As the patch size increases, the amount of information also increases. For example, the patch size of 45×45 covering an area of 2025 pixels has almost three times more information than the patch size of 27×27. Table 5.4 presents the evaluation metrics of S2 models with the stride size of 15×15 pixels and different patch sizes. From the patch size of 27 to 45, the mean rRMSE, mean relative bias and the coefficient of determination improve slowly. However, as the patch size keeps increasing significantly, these metrics turn around and start to decrease as an indication of worse fit of the model. This might suggest that having too much information has a negative effect on the prediction. The model S2-45,

which has the lowest mean rRMSE of 17.24% and the highest R^2 of 0.75, is selected as the baseline model for the subsequent experiments.

5.2 Quantifying the variation and reliability of predictions

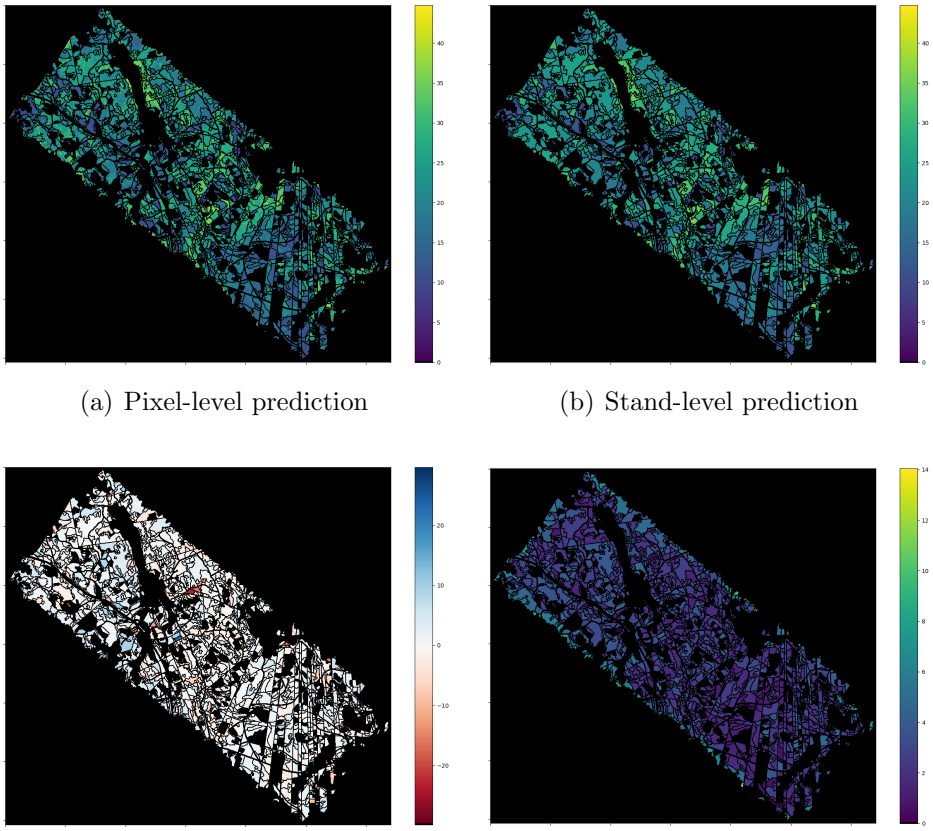
In this section, the baseline model S2-45 is used to predict every single pixels from HSI of the forest area and the predictions are quantified and analysed to answer the second research question. With over 155 million pixels in total, the HSI contains over 26 millions predictable pixels. To predict the forest labels of a pixel, a patch of size 45×45 pixels of its surrounding are extracted from the image to feed to the model. All the predicted pixels are attached together in the same position to create the whole prediction of the forest area.

Basal area is a forest variable to measure the above-ground surface area occupied by the alive trees in a forest stand and its unit is m^2/ha. Basal area is widely used in forestry to determine the tree density [3]. The higher basal area value implies that the tree-occupied area is larger. In this section, basal area is selected as the main forest variable for the following studies because it can be visibly verified with the RGB version of the hyperspectral image. In addition, the rRMSE of basal area in Table 5.2 is the second best result after woody biomass, which makes it reliable to study.

Figures 5.1(a) and 5.1(b) illustrate the pixel-level prediction and stand-level prediction of basal area, respectively. The forest variable prediction for each stand is calculated by averaging the predicted values of every pixel within it, therefore, the stand-level prediction is a smoothed version of the pixel-level ones. Figures 5.1(c) and 5.1(d) display the stand-level signed error and standard deviation of the basal area, respectively.

The stand-level signed error is the difference between the stand-level predicted and target values. In Figure 5.1(c), a majority of the stands are white or whitish, which means the differences between the predicted and target values are generally low. Near the center of the map in Figure 5.1(c), there are some red or blue signed-error stands, which indicates that the difference between the predicted and target values are higher than in the surrounding areas. These stands are examined closely in Figure 5.2.

The stand-level standard deviation is calculated from the pixel-level prediction in Figure 5.1(a). In Figure 5.1(d), the stands with the dark color have the standard deviation closer to zero. The dark shade shows that the pixel-level predicted values in that stand are clustered around the mean value.



(a) Pixel-level prediction

(b) Stand-level prediction

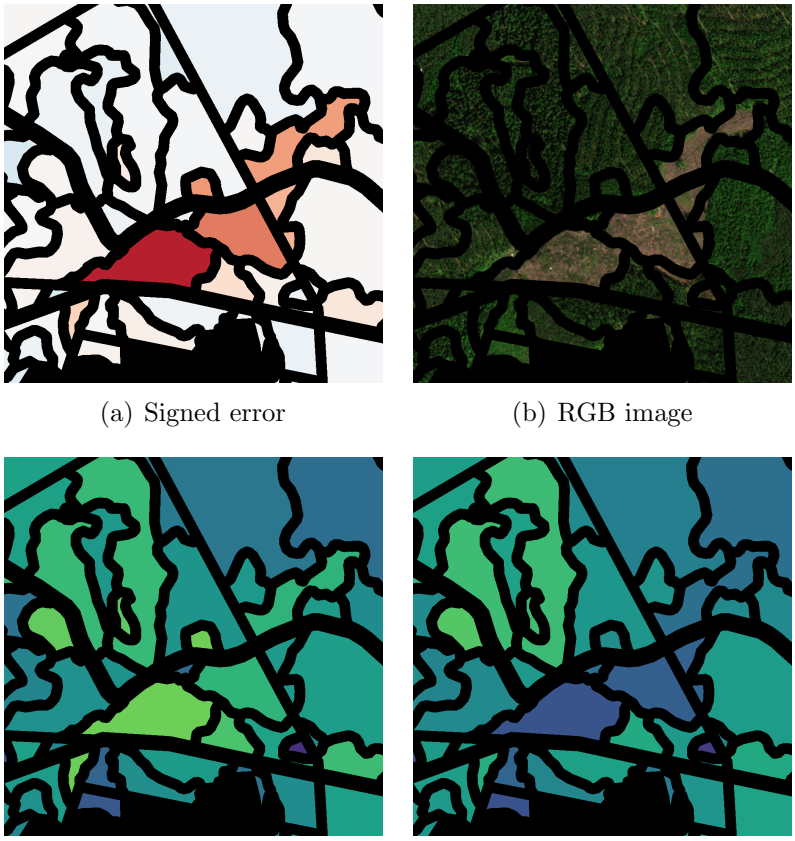
(c) Stand-level signed error

(d) Stand-level standard deviation

Figure 5.1: Basal area prediction maps

Those dark stands are generally in the center of the forest area, while the stands in the border area have much lighter colors. The border areas have higher standard deviation, hence, the pixel-level predicted values of the stand are more spread out. One potential explanation for these wide-ranging pixel-level predicted values is due to the lack of hyperspectral data when the patch size of 45×45 pixels extracted at the border. In Figure 3.1, the hyperspectral pixels that are out of the captured forest area are set as zeroes and shown as white.

Subsets of basal area maps are displayed in Figure 5.2 to study the stands with the highest signed error. The stand-level predicted values are lower while the target labels for those stands are much higher, meaning that the areas occupied by trees are larger as well. When cross-checking those target



(a) Signed error (b) RGB image

(c) Target labels (d) Stand-level prediction

Figure 5.2: Basal area detailed study 1.

labels with the RGB image, we find out that those stands are not covered with trees like their neighborhoods. It is contrasted with the target labels and similar to the stand-level predicted values. It is reasonable to conclude those target labels are set wrongly and the possible reason is due to the date the hyperspectral image was captured is not the same as the date the target labels were collected. Those stands affect negatively the evaluation metrics even though the predicted values are decent.

In Figure 5.3, another subsets of basal area maps are illustrated to study the stand with a high stand-level standard deviation. The examined stand is the greenish one in Figure 5.3(b). A high standard deviation stand suggests that a wide range of pixel-level predicted values in the stand. The target label and stand-level predicted label of that stand are equal while pixel-level predicted values of that stand in Figure 5.3(f) show blending of two

(a) RGB image (b) Stand-level standard deviation (c) 5×5 pixels standard deviation

(d) Target labels (e) Stand-level prediction (f) Pixel-level prediction

Figure 5.3: Basal area detailed study 2.

colors, light and dark. It suggests that there are two separate sets of pixel-level predicted values within this stand. The 5×5 pixel standard deviation in Figure 5.3(c) is dark with a thin blurry line in the middle, that can be understood as the border between the two sets of predicted values. When cross-checking the stand with the RGB image, the left half of the stand seems to be similar to the left neighbor stands while the right half of the stand is similar to the right neighbor stands. In this scenario, we can come to the conclusion that the high stand-level standard deviation comes from an inhomogeneous stand that could be separated into two smaller stands or merged with its adjacent stands. It also presents the reliability of the stand-level predicted value over pixel-level predicted values when the stand is inhomogeneous.

5.3 Scatter plots of stand-level predictions

The maps in Figure 5.1 are independent from each other, and it is difficult to draw any conclusions from them. Therefore, there are six basal area stand-level scatter plots in Figure 5.4 created from four different stand-level variables: *target*, *prediction*, *signed error* and *standard deviation*. The pairs of the two variables are formed by the same stand position and these scatter plots are used to study the correlations between the variables.

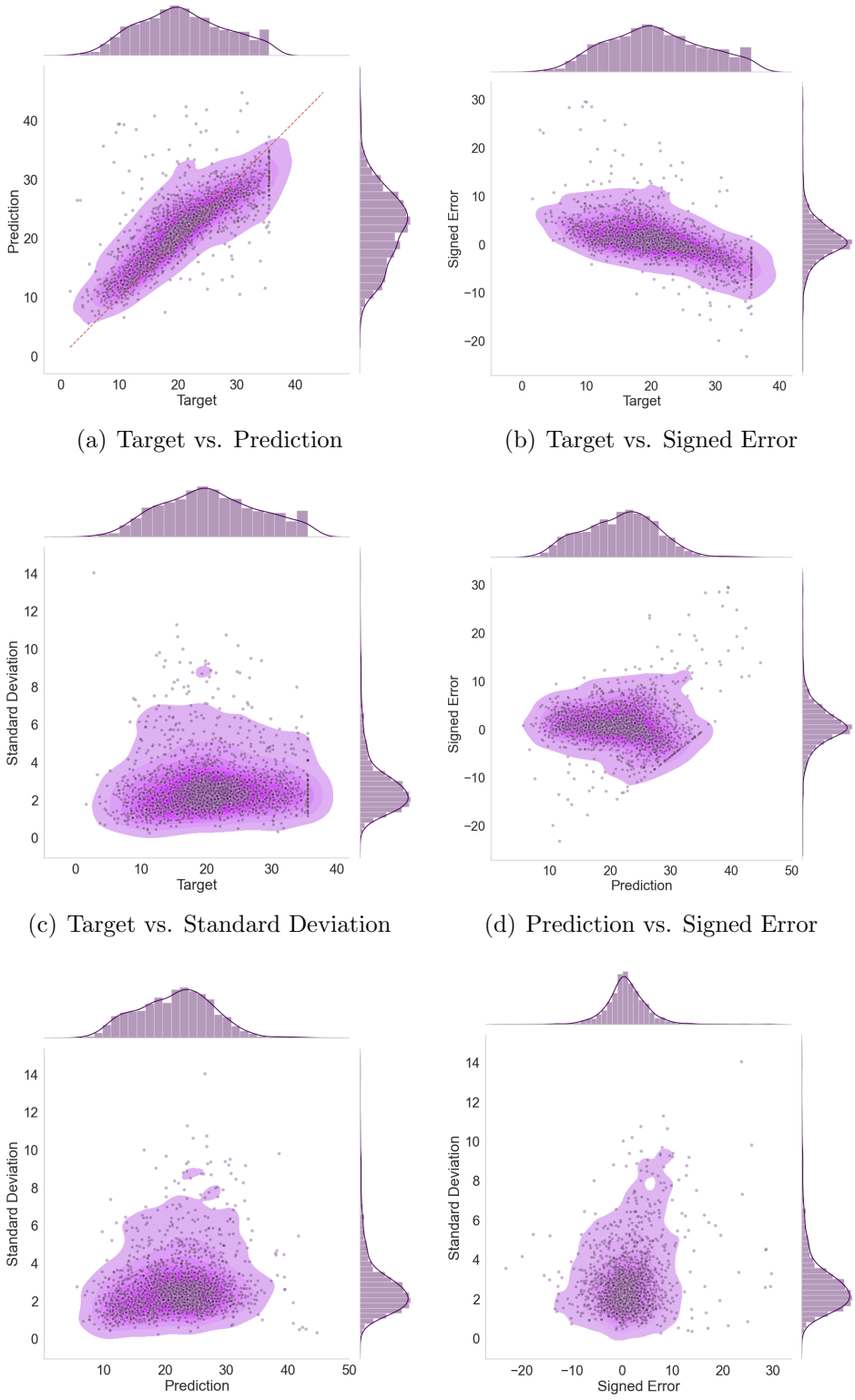
Figure 5.4(a) illustrates the scatter plot of the target value versus the prediction. The red diagonal line is the ideal prediction line where the prediction equals to the target value. The points are quite well-concentrated on the diagonal line where the prediction is correct. However, the prediction is falling below when the target value is greater than 20 m^2/ha and almost scattering around when the target value is over 30 m^2/ha. The maximum target value is around 35.5 m^2/ha, while the maximum prediction is 44.7 m^2/ha. Therefore, there is a small vertical line where the target value is maximum and the prediction varies from 20 to 30.

Figure 5.4(b) shows the plot of the target value versus the signed error. Signed error distribution looks like a bell curve with the mean of zero. The points are concentrated around the zero value of the signed error, but turning downward in the end where the target value is greater than 20 m^2/ha. The larger the target value is in basal area, the more negative the signed error is. It is consistent with the target value versus the prediction scatter plot in Figure 5.4(a) because the signed error is the difference between the prediction and the target value.

Figure 5.4(c) is the scatter plot of the target value versus standard deviation. Both the target value and the standard deviation are independent on each other. Most of the standard deviation values are in the range between zero and four and a majority of them are close to two. Two is a good standard deviation value for basal area because the mean of the stand-level target values is close to 22 m^2/ha.

Figures 5.4(d) and 5.4(e) are the scatter plots of the prediction versus the signed error and standard deviation, respectively. The prediction is quite independent from the signed error and standard deviation as both plots seem to have an additional noise or additive error instead of a multiplicative error. As signed error is quite constant at zero and standard deviation is constant at two, Figure 5.4(f) has a round and spherical distribution and the signed error and standard deviation seem to be independent from each other as well.

Figures 5.5, 5.6, 5.7 and 5.8 are the stand-level scatter plots of the remaining main continuous forest variables: *mean height*, *effective leaf area*



(a) Target vs. Prediction

(b) Target vs. Signed Error

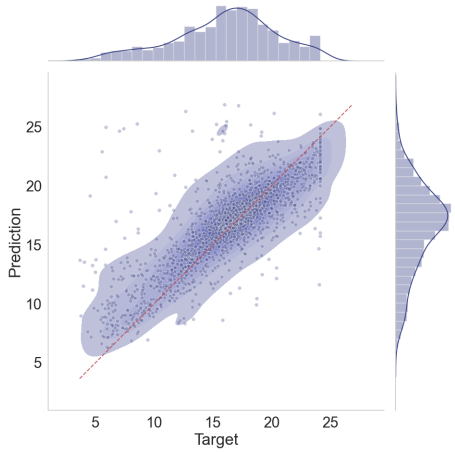
(c) Target vs. Standard Deviation

(d) Prediction vs. Signed Error

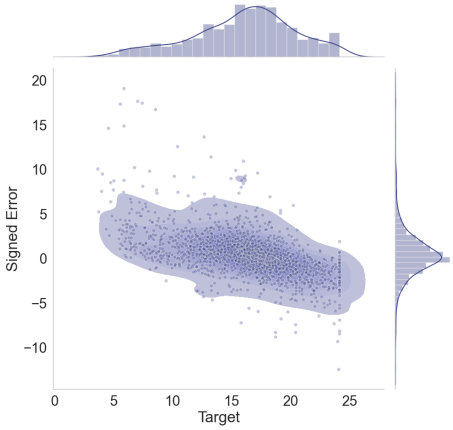
(e) Prediction vs. Standard Deviation

(f) Signed Error vs. Standard Deviation

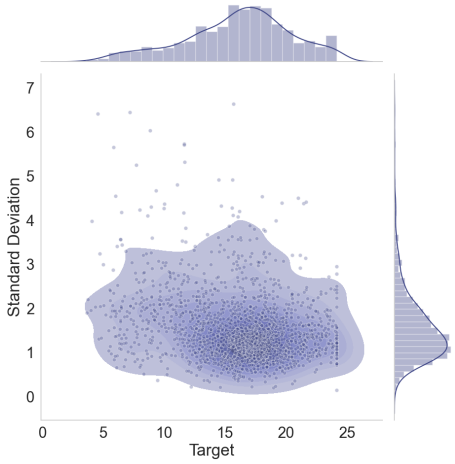
Figure 5.4: Basal area stand-level plots



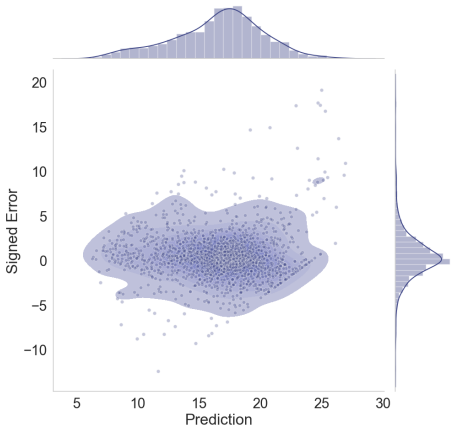
(a) Target vs. Prediction



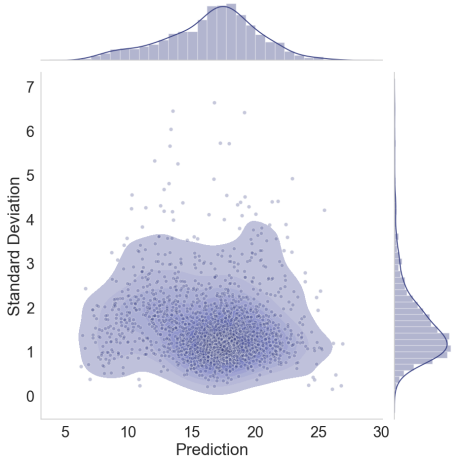
(b) Target vs. Signed Error



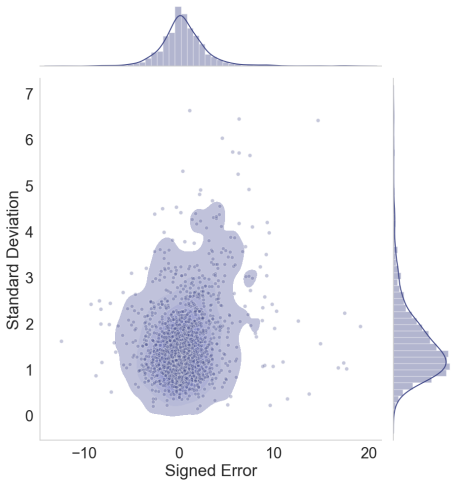
(c) Target vs. Standard Deviation



(d) Prediction vs. Signed Error

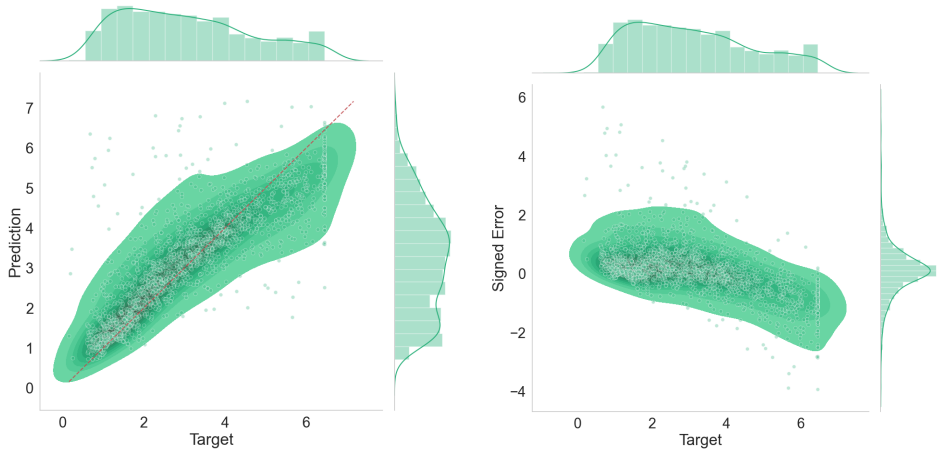


(e) Prediction vs. Standard Deviation



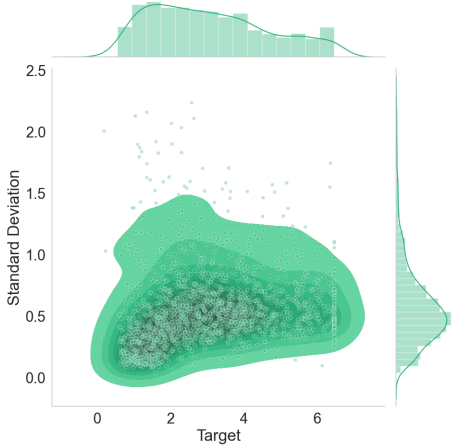
(f) Signed Error vs. Standard Deviation

Figure 5.5: Mean height stand-level plots

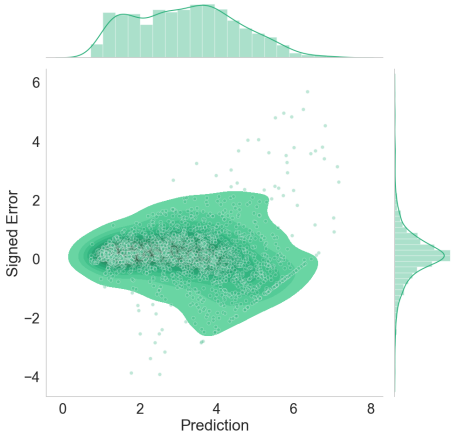


(a) Target vs. Prediction

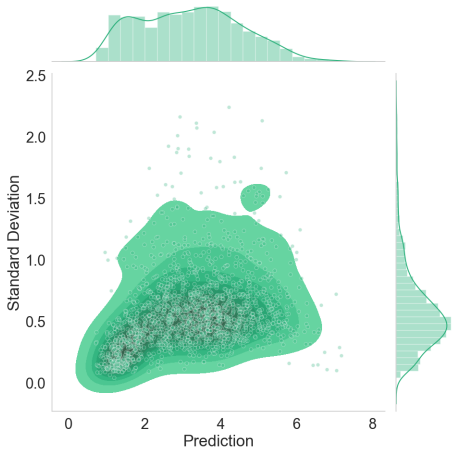
(b) Target vs. Signed Error



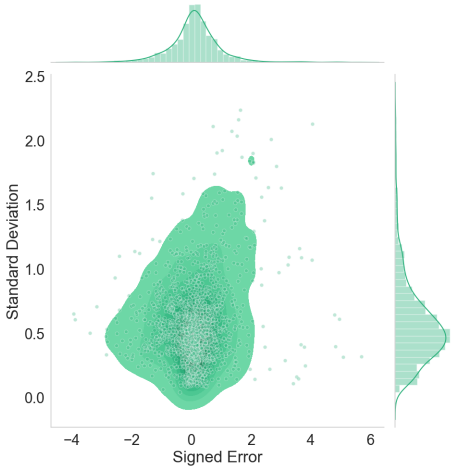
(c) Target vs. Standard Deviation



(d) Prediction vs. Signed Error

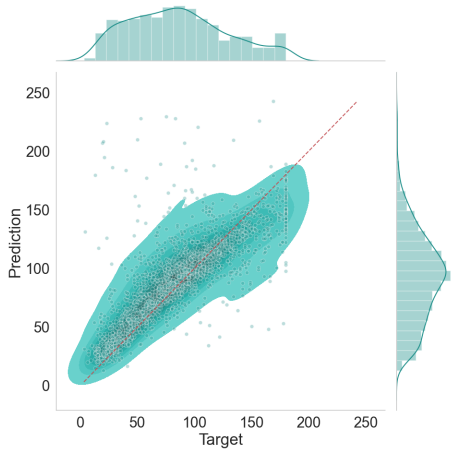


(e) Prediction vs. Standard Deviation

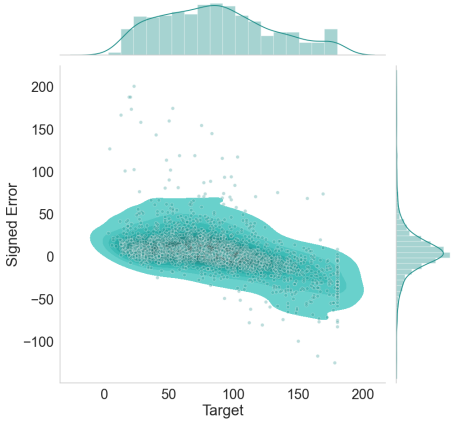


(f) Signed Error vs. Standard Deviation

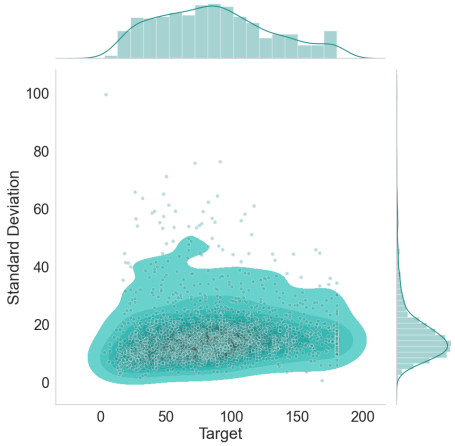
Figure 5.6: Effective LAI stand-level plots



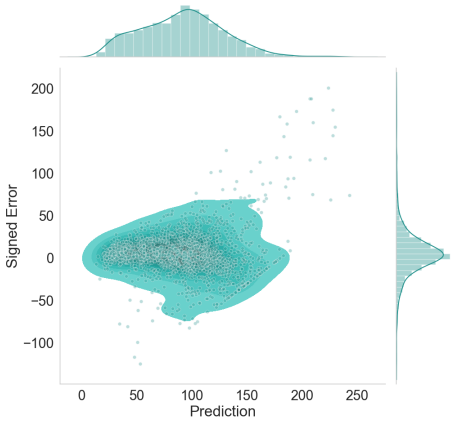
(a) Target vs. Prediction



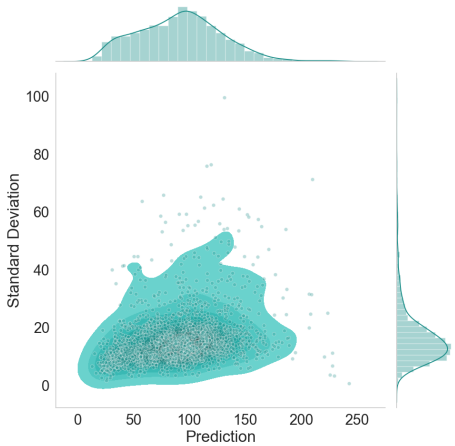
(b) Target vs. Signed Error



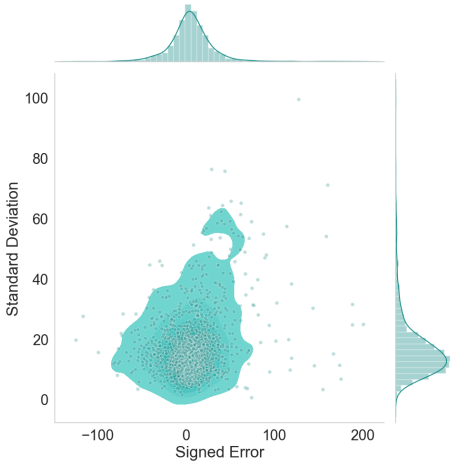
(c) Target vs. Standard Deviation



(d) Prediction vs. Signed Error

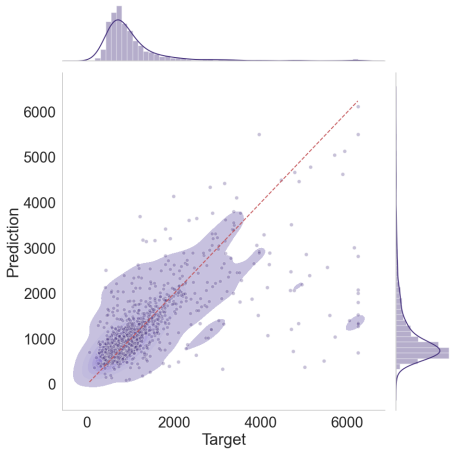


(e) Prediction vs. Standard Deviation

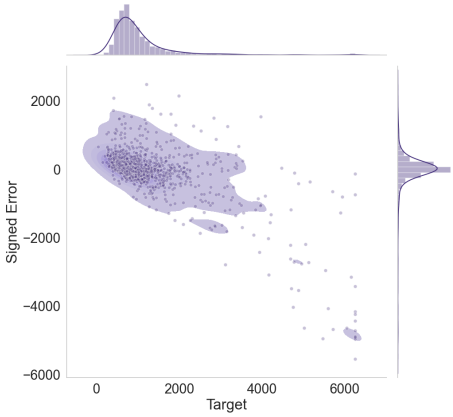


(f) Signed Error vs. Standard Deviation

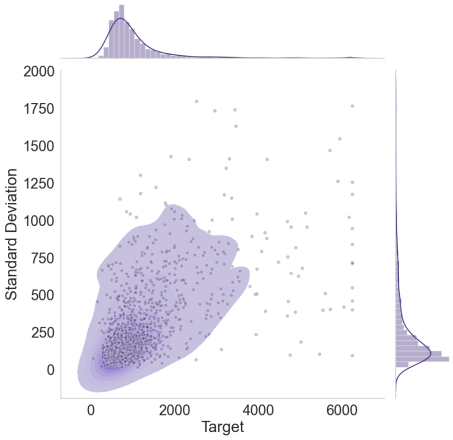
Figure 5.7: Woody biomass stand-level plots



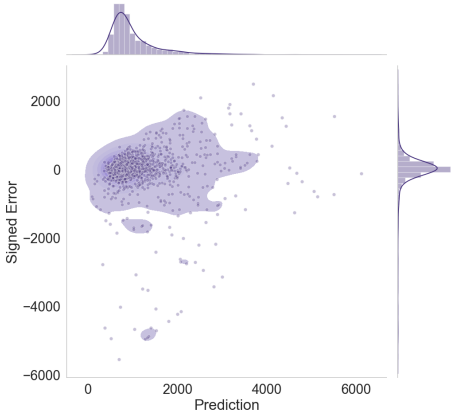
(a) Target vs. Prediction



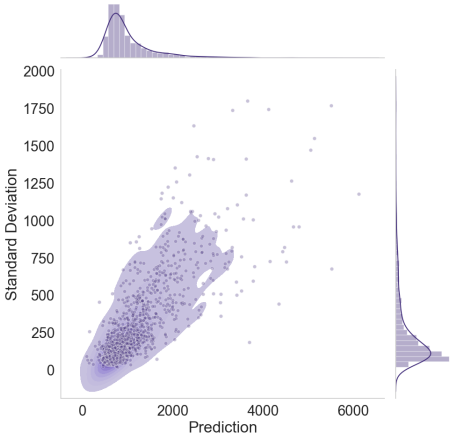
(b) Target vs. Signed Error



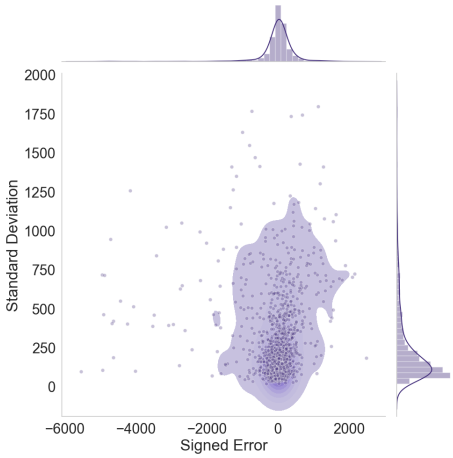
(c) Target vs. Standard Deviation



(d) Prediction vs. Signed Error



(e) Prediction vs. Standard Deviation



(f) Signed Error vs. Standard Deviation

Figure 5.8: Stem density stand-level plots

index, *woody biomass* and *stem density*, respectively. Out of the five forest variables, stem density has the worst prediction result with rRMSE is 39.94% and rBias is 5.57%. Woody biomass and effective leaf area index are in the same level with rRMSE around 24%, and basal area and mean height are the best ones in Table 5.2.

Mean height is the variable with the best evaluation results and it is also reflected in the mean height scatter plot of the target value versus the prediction from Figure 5.5(a) where the points are shown mostly on the diagonal line. Toward the end of the target value, the predictions of mean height are still in the line instead of falling short like the other three variables. In Figures 5.6(a), 5.7(a) and 5.8(a), effective leaf area index, woody biomass and stem density have larger amounts of small target values than basal area and, therefore, the distributions of them are not falling as low as basal area's one.

Similarly, the target values versus signed error scatter plots of mean height, effective leaf area index, woody biomass and stem biomass in Figures 5.5(b), 5.6(b), 5.7(b) and 5.8(b) have predictions falling below the targets when the target values are larger. The signed error distribution of those four forest variables look like a normal distribution with the mean of zero. However, the distribution curve of the signed error of stem density in Figure 5.8(b) is the narrowest and most peaked in the middle, from which it is inferred that the signed error distribution has smaller variance than its target distribution's variance.

The target values versus the standard deviation scatter plots of basal area, mean height, effective leaf area index and woody biomass in Figure 5.4(c), 5.5(c), 5.6(c) and 5.7(c) have the same distribution shape. The target value and standard deviation are independent from each other and the mean standard deviation value of each forest variable is around one tenth of its target value. However, the target values versus the standard deviation scatter plot of stem density is quite different. Stem density target distribution is right-skewed and has a very long tail where the maximum target value is six times larger than the mean target value. Stem density standard deviation distribution has the same shape, hence the distribution of the scatter plot has a linear form with a weak positive relationship. This can be interpreted as a result of a multiplicative error term instead of an additive error term.

The prediction versus signed error, the prediction versus standard deviation and signed error versus standard deviation scatter plots of mean height, effective leaf area index and woody biomass in Figures 5.5, 5.6 and 5.7 have the same characteristics as those of basal area described above. The prediction versus signed error and signed error versus standard deviation scatter plots of stem density are similar to the other forest variables, how-

Variable #	Variable name	Training			Test		
Categorical		micro	macro		micro	macro	
1	Fertility class	93.55 %	95.12 %		69.37 %	55.29 %	
2	Soil class	98.92 %	98.88 %		93.69 %	86.42 %	
3	Main tree species	96.59 %	97.62 %		87.39 %	83.67 %	
mean		96.36 %	97.20 %		83.48 %	75.13 %	
Continuous		rRMSE	rBias	R^2	rRMSE	rBias	R^2
4	**Basal area [m^2/ha]**	8.70 %	0.89 %	0.93	15.95 %	0.78 %	0.69
5	**Mean height [m]**	6.02 %	0.58 %	0.94	11.67 %	–0.92 %	0.77
6	**Woody biomass [t/ha]**	12.03 %	1.13 %	0.93	20.45 %	0.26 %	0.76
7	**Effective leaf area index**	10.90 %	1.44 %	0.95	21.29 %	0.91 %	0.81
mean		9.41 %	1.01 %	0.94	17.24 %	0.30 %	0.75

Table 5.5: Evaluation result of all forest variables obtained from training and testing stands.

ever, the prediction versus standard deviation scatter plot of stem density in Figure 5.8(e) is quite different and looks similar to its target versus standard deviation scatter plot. It also has a linear form with a strong positive correlation. Again, this hints that the error term is multiplicative.

5.4 Comparing predictions obtained for training and test stands

In the last experiment, the baseline model S2-45 was used to compare the predictions obtained for the training and test sets. The results are presented in Table 5.5. The categorical variable prediction results for the training set are significantly better than for the test set as the overall micro accuracy is 13 %-units higher and mean class macro accuracy is 22 %-units higher. These results indicate that the model is overfitting for the training set. Similarly, the continuous variable prediction results also indicate that the mean rRMSE of the training set is 9.41%, which is much lower than the 17.24% of the test set. The coefficient of determination, or the fitness, of the training set almost reaches to one while for the test set it is only 0.75. The relative bias of both the training and test sets are alike. In addition to these results, the overlap percentage between the training and test sets in Figure 3.5 is only 0.60%, and it shows that the model is only overfitting for the training set and not for the test set as their areas are not overlapping.

During the training process for the baseline model, a snapshot of the model is saved after every epoch (1200 steps). Figure 5.9 is a line chart that displays the stand-level categorical accuracy for training and test sets during the training process. The overall accuracy (micro) and mean class accuracy

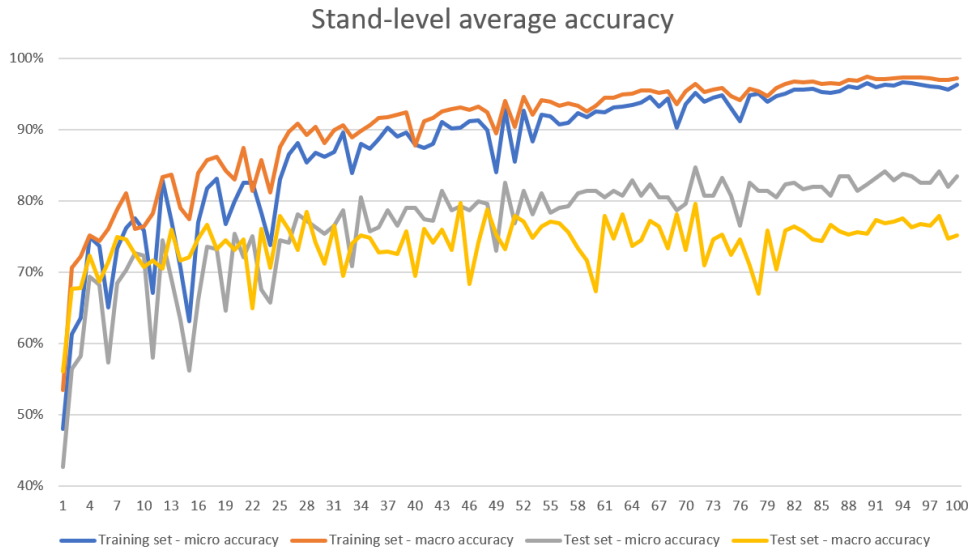


Figure 5.9: The stand-level average categorical accuracy during the training process.

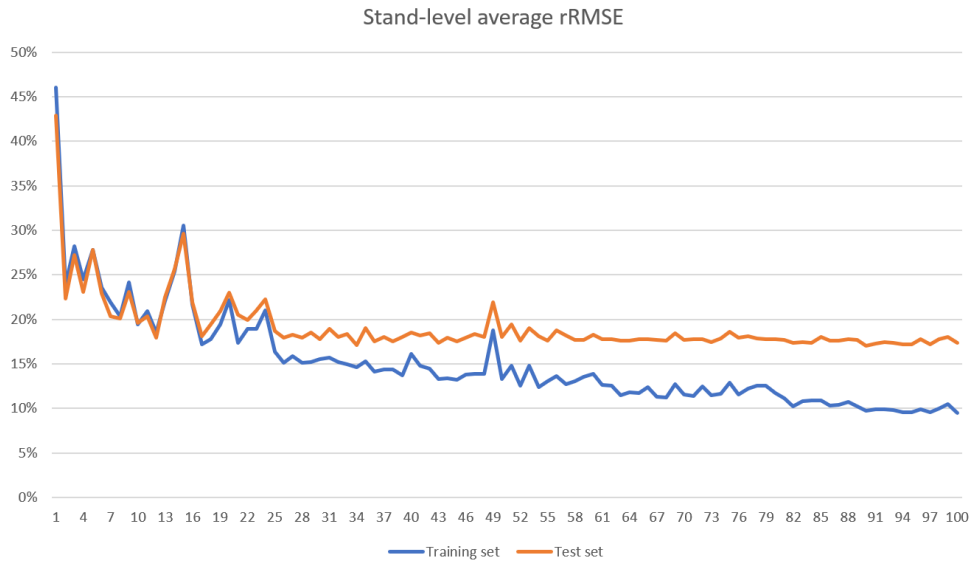


Figure 5.10: The stand-level average rRMSE of the continuous variables during the training process.

(macro) for each set are recorded in stand-level after every epoch.

The micro accuracy of both training and test sets starts at around 45 to 50% and improves drastically during the first 30 epochs to 86 and 75%, respectively. They continue to improve at a slower pace to 92 and 82% during the next 30 epochs. The micro accuracy of the training set increases slightly while the micro accuracy of the test set fluctuates until the end of the training process. In the first 15 to 20 epochs, the accuracy difference between the two sets is consistently between five to seven. After that, it is expanding rapidly to ten as the model training progresses to the 30-th epoch and slowly again for the rest of the training process.

The macro accuracy of the training set increases similarly to the micro one. However, the progress of the macro accuracy for the test set is quite different. It grows quickly to 75% within the first 30 epochs and starts to oscillate in a wide range of 10 %-units for the next 50 epochs. It only stabilizes at around 77% during the end of the training.

Figure 5.10 illustrates the average rRMSE of the continuous variables for the training and test sets during the training process. The average rRMSE is only calculated from the four main continuous variables: basal area, mean height, woody biomass and effective leaf area index. During the first 25 epochs, the average rRMSE of both sets are almost overlapping each other and drops sharply from 45 to 22%. From there onward, the difference between the average rRMSE values is expanding constantly. The average rRMSE of the test set decreases slowly to 17% at the 50-th epoch and fluctuates around that value for the rest of the training process. Meanwhile, the average rRMSE of the training set decreases more to 9.41% and might keep going down if the training process were continued.

Both analyses clearly show that the model has started overfitting for the training set during the training process. This is a natural phenomenon in training of the deep neural models. Typically, the ramification of overfitting is poor performance on unseen data. However, the analyses also show that keeping training not only improves the accuracy for the training set but also for the test set. When the accuracy for the test set does not make progress anymore, the model hits its limit despite the improvement of the accuracy for the training set.

Chapter 6

Conclusions

A novel convolutional neural network (CNN) has been developed in the AIROBEST project to predict multiple forest variables simultaneously. Despite of good results for both categorical and continuous tasks, there are some challenges related to the pixel-level prediction results and the way the hyperspectral forest data was used. The goal of this thesis was to tackle those challenges and evaluate the predictions of the CNN model from the stand-level perspective.

In this thesis, a new procedure for data processing and data split was introduced. The procedure utilizes TAIGA hyperspectral forest data to produce a larger amount of samples for both training and test sets while maintaining the low overlap percentage between the sets. The model family S2 with the stride size of 13×13 pixels was selected as it balances between the number of samples and the duration of the model training. In addition, a new set of metrics was proposed and studied to evaluate the prediction results for the stand-level predictions. rRMSE and R^2 are the key metrics for the continuous tasks and overall micro accuracy and mean class macro accuracy are the key metrics for categorical tasks.

By using the new evaluation metrics, we answered the first research question about the difference between the pixel and stand-level data split. The experiments showed that the models that use stand-level data split to train have much better result than others. Continuing using the new metrics to evaluate the predictions, we discovered that the model S2-45 has the best result overall. The model S2-45 achieved the best results with the lowest mean rRMSE of 17.24% and the highest R^2 of 0.75 for the continuous tasks as well as overall accuracy of 83.48% and mean class accuracy of 75.13% for the categorical tasks. The CNN architecture designed for the pixel-level prediction is thus still working well for the stand-level prediction.

The model S2-45 was used to predict every pixel's forest variables from

HSI and to create the prediction maps, such as target, stand-level prediction, stand-level signed error and stand-level standard deviation. By studying the maps, we highlighted some issues within TAIGA reference data. These issues included that some target labels are set wrongly or some stands are inhomogeneous and could be separated into smaller stands. We also managed to quantify the variation and reliability of the predictions from the scatter plots of the stand-level predictions. The last experiment showed that the CNN model is overfitting for the training set with the new procedure of data processing and splitting. Despite of that, the CNN model still performed well on the test set as the prediction result improved during the training process.

In summary, we introduced a new set of metrics and used it to evaluate the stand-level predictions. Even though that meets the goal of this thesis, there exists some room for improvement and future development. Firstly, both the identified inaccuracy issues within TAIGA reference data jeopardize the training and the prediction. They can be fixed to train a new model. Secondly, there is an opportunity to explore the relationship between the overlap percentage of the training and test sets and the prediction results. Lastly, instead of using 671 stands, we could use all the stands to generate the samples. The evaluation result might not be comparable with other trained models' result, but it would reflect the whole TAIGA hyperspectral forest data.

Bibliography

[1] Bandos, T. V., Bruzzone, L., and Camps-Valls, G. Classification of hyperspectral images with regularized linear discriminant analysis. *IEEE Transactions on Geoscience and Remote Sensing 47*, 3 (2009), 862–873.

[2] Baumgardner, M. F., Biehl, L. L., and Landgrebe, D. A. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3, Sep 2015.

[3] Bettinger, P., Boston, K., Siry, J. P., and Grebner, D. L. Chapter 2 - valuing and characterizing forest conditions. In *Forest Management and Planning (Second Edition)*, P. Bettinger, K. Boston, J. P. Siry, and D. L. Grebner, Eds., second edition ed. Academic Press, 2017, pp. 21–63.

[4] Bioucas-Dias, J., Plaza, A., Camps-Valls, G., Scheunders, P., Nasrabadi, N., and Chanussot, J. Hyperspectral remote sensing data analysis and future challenges. *Geoscience and Remote Sensing Magazine, IEEE 1* (06 2013), 6–36.

[5] Boldrini, B., Kessler, W., Rebner, K., and Kessler, R. Hyperspectral imaging: A review of best practice, performance and pitfalls for in-line and on-line applications. *Journal of Near Infrared Spectroscopy 20* (10 2012), 438–.

[6] Caruana, R. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning* (1993), Morgan Kaufmann, pp. 41–48.

[7] Chen, Y., Jiang, H., Li, C., Jia, X., and Ghamisi, P. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing 54*, 10 (2016), 6232–6251.

[8] Chen, Y., Lin, Z., Zhao, X., Wang, G., and Gu, Y. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7*, 6 (2014), 2094–2107.

[9] Chen, Y., Zhao, X., and Jia, X. Spectral–spatial classification of hyperspectral data based on deep belief network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 8*, 6 (2015), 2381–2392.

[10] Chollet, F. *Deep Learning with Python*, 1st ed. Manning Publications Co., USA, 2017.

[11] Dahl, G. E., Sainath, T. N., and Hinton, G. E. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), pp. 8609–8613.

[12] Goodfellow, I. J., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[13] Halme, E. Utility of remotely sensed spectral information in forest variable estimation. Master's thesis, Aalto University. School of Engineering, 2018.

[14] Halme, E., Pellikka, P., and Mõttus, M. Utility of hyperspectral compared to multispectral remote sensing data in estimating forest biomass and structure variables in Finnish boreal forest. *International Journal of Applied Earth Observation and Geoinformation 83* (2019), 101942.

[15] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9* (12 1997), 1735–80.

[16] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167* (2015).

[17] Kayes, I., and Mallik, A. *Boreal Forests: Distributions, Biodiversity, and Management*. Springer International Publishing, Cham, 2020, pp. 1–12.

[18] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc.

[19] Labate, D., Safari, K., Karantzas, N., Prasad, S., and Shahraki, F. F. Structured receptive field networks and applications to hyperspectral image classification. In *Wavelets and Sparsity XVIII* (2019), D. V. D. Ville, M. Papadakis, and Y. M. Lu, Eds., vol. 11138, International Society for Optics and Photonics, SPIE, pp. 218 – 226.

[20] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[21] Lecun, Y., Haffner, P., and Bengio, Y. Object recognition with gradient-based learning.

[22] Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., and Benediktsson, J. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing PP* (04 2019), 1–20.

[23] Licciardi, G., Marpu, P., Chanussot, J., and Benediktsson, J. Linear versus nonlinear pca for the classification of hyperspectral data based on the extended morphological profiles. *Geoscience and Remote Sensing Letters, IEEE 9* (05 2012), 447–451.

[24] Mou, L., Ghamisi, P., and Zhu, X. X. Deep recurrent neural networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing 55*, 7 (2017), 3639–3655.

[25] Mõttus, M., Molinier, M., Halme, E., Cu, H., and Laaksonen, J. Patch size selection for analysis of sub-meter resolution hyperspectral imagery of forests.

[26] NAGELKERKE, N. J. D. A note on a general definition of the coefficient of determination. *Biometrika 78*, 3 (09 1991), 691–692.

[27] Nair, V., and Hinton, G. Rectified linear units improve restricted boltzmann machines vinod nair. vol. 27, pp. 807–814.

[28] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning, 12 2020.

[29] Pham, P. Deep learning methods for modelling forest biomass and structures from hyperspectral imagery. Master's thesis, Aalto University School of Science, 2019.

[30] Quiza, R., and Davim, J. *Computational Methods and Optimization*. 01 2011, pp. 177–208.

[31] Ruder, S. An overview of multi-task learning in deep neural networks. *CoRR abs/1706.05098* (2017).

[32] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning Representations by Back-propagating Errors. *Nature 323*, 6088 (1986), 533–536.

[33] S. Mohamed, I. *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques*. PhD thesis, 09 2017.

[34] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization?, 2019.

[35] Schmidhuber, J. Deep learning in neural networks: An overview. *CoRR abs/1404.7828* (2014).

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 56 (2014), 1929–1958.

[37] Villa, A., Benediktsson, J. A., Chanussot, J., and Jutten, C. Hyperspectral image classification with independent component discriminant analysis. *IEEE Transactions on Geoscience and Remote Sensing 49*, 12 (2011), 4865–4876.

[38] Wang, S., Wang, Q., and Gong, M. Multi-task learning based network embedding. *Frontiers in Neuroscience 13* (01 2020).

[39] Yin, W., Kann, K., Yu, M., and Schütze, H. Comparative study of CNN and RNN for natural language processing. *CoRR abs/1702.01923* (2017).

[40] Zeiler, M. D., Ranzato, M., Monga, R., Mao, M. Z., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. E. On rectified linear units for speech processing. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), 3517–3521.